commodore

plus/4

FELHASZNÁLÓI KÉZIKÖNYV

PLUS/4-es

Felhasználói kézikönyv

NOVOTRADE · 1986

A mű eredeti címe: Commodore/Plus/4 User Manual (1984)
Fordította: Magyarics Péter és Varga Antal

A kiadásért felel: RÉNYI GÁBOR, a NOVOTRADE RT igazgatója
Kiadványmenedzser: BÉKÉS TAMÁS
Felelős szerkesztő: TARR KÁLMÁNNÉ
Műszaki szerkesztő: Hajdú Árpád

Szedte: Nyomdaipari Fényszedő Üzem (867747/09)
Készült Gutenberg Nyomdaipari Kisszövetkezet Nyomdájában – 86/00[illegible]
Felelős vezető: Óvári László
Budapest, 1986

Készült a Gutenberg Nyomdaipari Kisszövetkezet Nyomdájában — 86/0075
Felelős vezető: Óvári László
Budapest, 1986
és a Fejér Megyei Nyomdaipari Vállalat székesfehérvári üzemében
Felelős vezető: Gyenti Pál igazgató
86/1655777

ISBN 963 02

© Hungarian translation Magyarics Péter, Varga Antal
Copyright C 1984 Commodore Electronics Ltd

Minden jog fenntartva. A Commodore cég írásbeli hozzá¡árulása nélkül tilos a könyvet vagy annak részeit bármilyen eljárással (nyomtatás, fotokópia vagy egyéb technika), elektronikus rendszerek felhasználásával másolni, sokszorosítani, terjeszteni

Tartalomjegyzék

Bevezetés 5

1. fejezet: A teljes rendszer 9
   Ismerkedés a kapcsolókkal és csatlakozásokkal 9
   A Commodore PLUS/4 üzembe helyezése 11
   Hibaelhárítás 12
   Perifériák 12
2. fejezet: A billentyűzet és a képernyő használata 14
   A billentyűzetről általában 15
   A speciális billentyűzet 15
   Grafikus billentyűk 18
   Programbillentyűk (F1...F8) 19
   A HELP használata 20
3. fejezet: Szoftverhasználat 21
   Bevezetés 22
   Beépített szoftver 22
   Modulok 22
   Kazetták 23
   Mágneslemez 24
4. fejezet: A programozás első lépései 28
   Bevezetés 29
   Színek a billentyűzeten 29
   Színes és inverz kiíratás 30
   Néhány egyszerű program 31
   Hibajavítás 32
   Bevezetés a képernyő használatába 34
   Képernyőre való íratás részletesebben 35
   Képernyő ablakok használata 37
5. fejezet: Számok és számítások 38
   Számok és műveletek 39
   Műveletek számokkal 40
   Változók 42
   Numerikus függvények 43
   A véletlen számot előállító függvény 44
6. fejezet: Ismerkedés a BASIC programozási nyelvvel 46
   Bevezetés 47
   Programozási módok 47
   Beviteli/kiviteli utasítások (I/O utasítások) 48
   Vezérlő utasítások és ciklusok 51

Feltételes utasítások ... 53
Szubrutinok ... 53
Megjegyzés (REM) ... 54
7. fejezet: A grafika és a színek használata ... 56
Grafikus karakterek ... 57
Karaktermozgatás ... 59
A színek kezelése ... 61
Finomfelbontású grafika ... 62
Pontok, vonalak, feliratok ... 64
Négyszögek, körök, sokszögek, festés ... 65
Többszínű grafika ... 69
8. fejezet: Hang és zene programozása ... 71
Bevezetés ... 72
A hangerő vezérlése ... 72
Zenei hang megszólaltatása ... 72
Zenei hanghatások ... 73
Zeneszerzés ... 74
A PLUS/4 mint "zenegép" ... 76
BASIC Ismerettár ... 79
1. rész: Bevezetés ... 80
A parancsok és utasítások formája ... 81
BASIC parancsok ... 82
BASIC utasítások ... 90
További ismeretek a grafika használatáról ... 109
Függvények ... 110
Numerikus függvények ... 110
Sztringkezelő függvények (karakterfüggvények) ... 113
Egyéb függvények ... 114
Változók és műveleti jelek ... 115
BASIC műveletek ... 118
2. rész: BASIC 3.5 rövidítések ... 119
3. rész: Fordítóprogramok ... 122
4. rész: Hibaüzenetek ... 124
5. rész: TEDMON ... 130
6. rész: Képernyőkódok ... 136
7. rész: ASCII és CHR $ kódok ... 139
8. rész: Képernyőtár és színtár térképek ... 142
9. rész: Tárregiszter térkép ... 144
10. rész: Matematikai függvények kiszámítása ... 145
11. rész: Zenei hangjegy táblázat ... 146
12. rész: Kipróbálandó programok ... 148
13. rész: RS–232 Interface ... 151

Bevezetés

Nagyszerűen választott! A Commodore PLUS/4 az első olyan személyi számítógép, amelyet valóban otthoni alkalmazásra készítettek. Természetesen minden olyan egyéb feladat elvégzésére is alkalmas, amely egy házi számítógéptől elvárható. Ennek a kézikönyvnek az a célja, hogy az olvasó megtanulja mindazokat a „dolgokat", amelyek a PLUS/4-gyel elvégezhetők:

- Összeállítani és üzembehelyezni a gépet
- A billentyűzetről elvégezhető különböző feladatokat végrehajtani
- A különböző szoftvereket használni
- A PLUS/4 matematikai, grafikus, hang és programozási lehetőségeit kihasználni

Ha az olvasó már rendelkezik valamennyi programozási előismerettel, és elsősorban a Commodore PLUS/4 másik kézikönyve – A PLUS/4 szoftver kézikönyv – iránt érdeklődik, akkor is ajánlatos ennek a könyvnek az áttanulmányozása.

Melyek a PLUS/4 különleges tulajdonságai?

64 k RAM (Írható és olvasható tár, amelyből 60 k használható fel a BASIC nyelvű programokhoz)
Teljes írógép-rendszerű billentyűzet
Beépített szoftver.
Képernyő-ablak szerkesztés.
HELP billentyű.
8 előre programozott és újraprogramozható program-billentyű.
4 különálló kurzormozgató billentyű.
A legtöbb C64 és VIC–20 perifériakus egység felhasználható.
121 szín (15 alapszín, 8 élességszint + fekete).
75-nél több BASIC utasítás
Finomfelbontású grafika.
Szövegírás és finomfelbontású grafika, osztott képernyőn.
Grafikus karakterkészlet a billentyűzeten.
Színbillentyűk.
320 × 200-as képernyőfelbontás.
Inverz és villogó karakterek.
Két-tónusú hanggenerátor.
Beépített gépi kód (Monitor, 17 utasítás).

1. FEJEZET

A teljes rendszer
Ismerkedés a kapcsolókkal és csatlakozásokkal
A Commodore PLUS/4 üzembehelyezése
Hibaelhárítás
Perifériák

A teljes rendszer

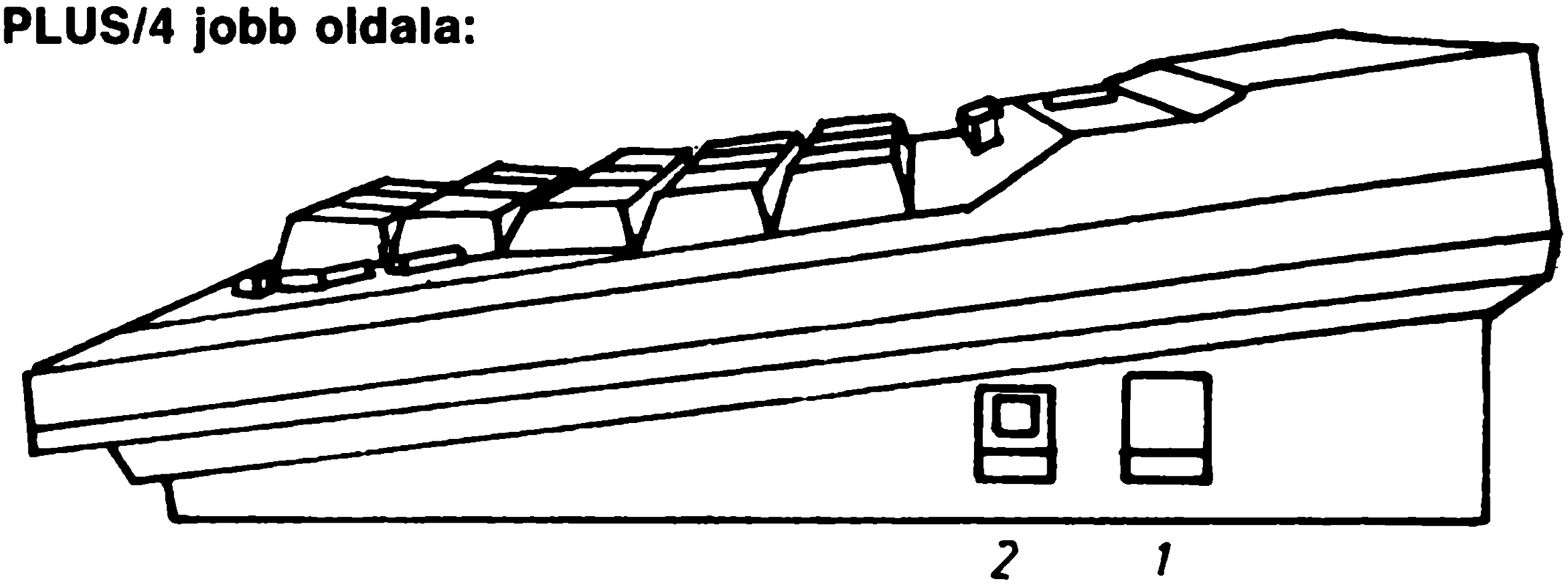
Miután kinyitottuk a számítógépet tartalmazó dobozt, ellenőrizzük, hogy az alábbi lista szerint minden tétel megtalálható-e?

1) A Commodore PLUS/4 számítógép
2) A tápegység. A kábel egyik vége a számítógép hátoldali csatlakozóaljzatába, a másik pedig a fali dugaszolóaljzatba illeszkedik.
3) RF kábel. Ez köti össze a televíziót a PLUS/4 bal oldalán lévő, RF jelzésű kimeneti jack-dugóval. (Ha monitort használunk, erre a kábelre nincs szükség.)

Mielőtt bármilyen csatlakoztatást elvégzünk, tanulmányozzuk a kézikönyvben a következő ábrákat. Ezeken az összes csatlakozópont látható, így könnyen és gyorsan összeállíthatjuk a teljes rendszert.

Ismerkedés a kapcsolókkal és csatlakozásokkal

A PLUS/4 jobb oldala:



1. hálózati kapcsológomb

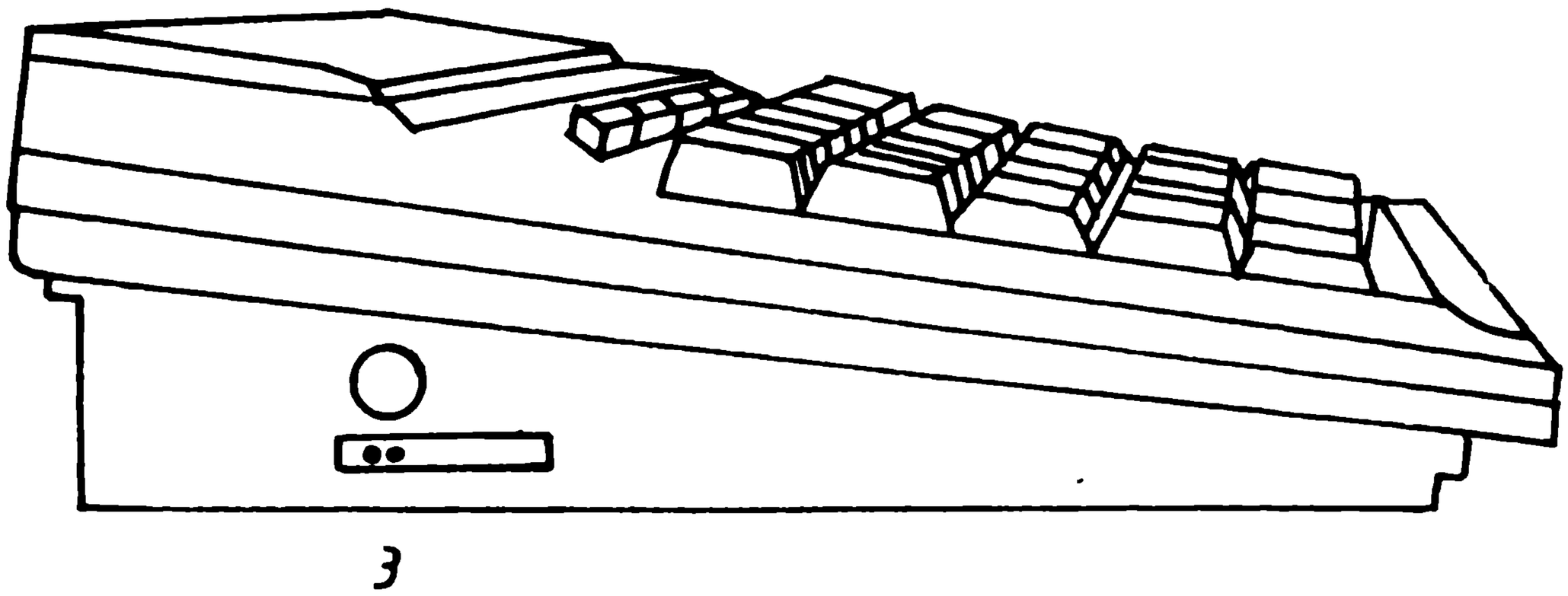
A számítógép kikapcsolt (OFF) helyzetben legyen, amikor csatlakoztatjuk vagy eltávolítjuk a perifériákat (kazettás egység, nyomtató stb.). A billentyűzet alatt a bal oldalon a piros hálózati kapcsolásjelző fény a tápfeszültség bekapcsolását jelzi.

2. RESET kapcsoló. Ezt a gombot kétféle módon használhatjuk.

a) Bekapcsolás után, működés közben, ha benyomjuk azt a gombot, törlődik a tárból a BASIC nyelvű program, a gép alaphelyzetbe kerül.
b) Ha a gépet úgy akarjuk alapállapotba hozni, hogy a tárban BASIC nyelvű programok is megmaradjanak, a RUN/STOP billentyű benyomott helyze-

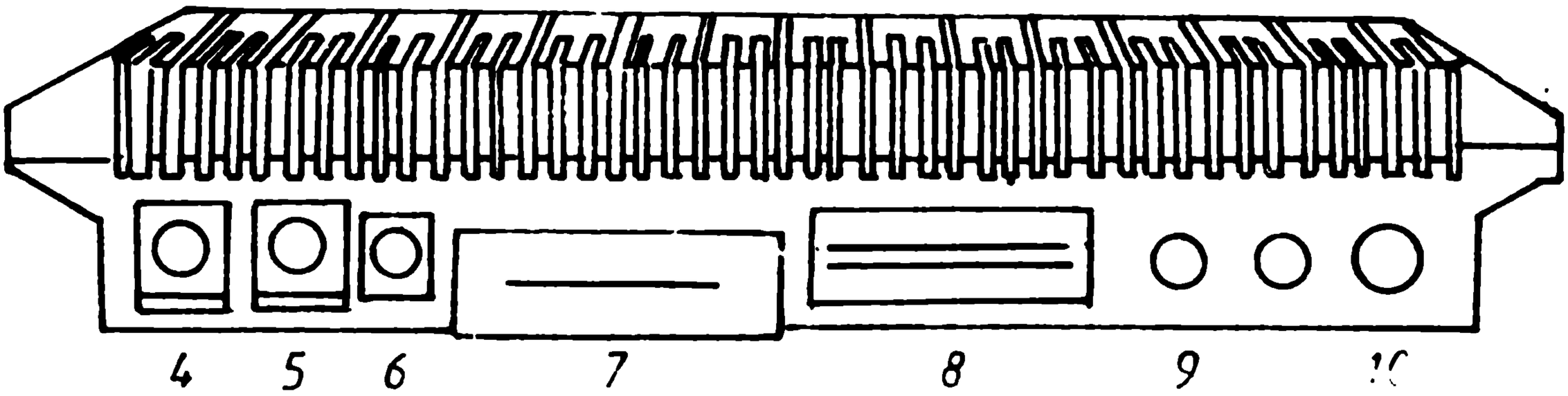
tében nyomjuk meg a RESET gombot. A gép ekkor áttér a beépített gépi nyelvű monitorra. (Ennek használatára most nem térünk ki.) Ha BASIC-re akarunk visszatérni, írjunk be egy X-et, majd nyomjuk le a RETURN billentyűt. Így a BASIC programok érintetlenek maradnak. Ha a képernyőre akarunk kiíratni egy programot, gépeljük be a LIST parancsot, majd a RETURN billentyűt – mint minden parancs után – nyomjuk be.

A PLUS/4 bal oldala:



Az itt látható – az ábrán 3-mal jelölt – RF aljzatba a tv csatlakoztatható. Ide helyezzük be a vékony fekete kábel egyik végét, a másikat pedig a tv-készülék antenna-bemenetére.

A PLUS/4 hátoldala:



Az itt látható aljzatok számos csatlakoztatási lehetőséget nyújtanak, mindegyik más célt szolgál. Vigyázzunk, hogy mindig a megfelelőt használjuk! A csatlakozások a következők:

4. Hálózati csatlakozó aljzat:
   Ide csatlakozik a hálózati kábel. Először mindig itt illesszük össze, és utána a fali csatlakozásnál!
5. Soros csatlakozó aljzat (Serial):
   Ezen keresztül illeszthetjük a lemezmeghajtót vagy a nyomtatót. Ha mindkettőt egyszerre akarjuk használni, akkor ebbe az aljzatba a lemezmeghajtóval csatlakozzunk, és annak a hátoldaláról a nyomtatóba.
6. Kazettásegység csatlakozó:
   Csak C=1531 Datasette típusú magnetofont használhatjuk.

7. User-port (RS–232)
   Ide illeszthetünk különböző tartozékokat, pl. az RS–232 adaptert. Ez az adapter lehetővé teszi olyan tartozékok használatát is, amelyek nem szabványos Commodore aljzatokon keresztül csatlakoztathatók.
8. Tárbővítő:
   A PLUS/4 szoftver kazettáinak behelyezésére való. Mielőtt betesszük vagy kivesszük, ellenőrizzük, hogy a gép ki van-e kapcsolva!
9. Botkormány (joystick) csatlakozások (játék-csatlakozások): Ebbe a két aljzatba csatlakoznak a botkormányok. A PLUS/4 speciálisan tervezett botkormányokkal működik.
10. Video-bemenet:
    Ide csatlakoztatjuk a monitor kábelét. Noha ez az aljzat 8 pólusú, használhatunk 5 pólusú kábelt is!

A Commondore PLUS/4 üzembe helyezése

Figyelem! Az összeállításhoz legalább két fali csatlakozóaljzatra van szükség: egy a számítógépnek, egy pedig a tv-készüléknek.
Ha lemezmeghajtót vagy nyomtatót is használunk, további csatlakozási lehetőségek szükségesek.
A számítógépet a tv-készüléktől kényelmes távolságra kell elhelyezni.
Ellenőrizzük, hogy a rendszer összeállításakor a gép kikapcsolt állapotban van, és a hálózati ellenőrzőfény nem világít.

Az üzembehelyezés menete:

1. Ezután csatlakoztassuk a tv-kábel egyik végét a tv-készülék antenna-csatlakozójába, a másikat pedig a számítógép RF jelzésű aljzatába.
   Megjegyzés: ha a PLUS/4-et nem tv-vel, hanem monitorral használjuk, kövessük a monitor kézikönyvben leírtakat. Ebben az esetben a hátlapon található VIDEO kimenetet használjuk.
2. A gép és a tv összekapcsolása után a hálózati csatlakozókábelt illesszük a PLUS/4-hez. A hátsó oldalon levő aljzatba a kábel szögletes végződése illik, a másik vége a fali dugaszoló aljzathoz csatlakozik.
3. Kapcsoljuk be a tv-készüléket, és a PLUS/4-et (ON). A tv-n válasszunk ki egy használaton kívüli csatornát, és az állomáskeresőt állítsuk a 36-osra. Ha minden rendben van, a következő üzenet jelenik meg a képernyőn:

```
COMMODORE BASIC 3.5 60671 BYTES FREE
READY
```

A READY alatti villogó jel, a kurzor arról tájékoztat, hogy a számítógép készen áll a működésre. A beírt betűk feketén jelennek meg a fehér háttérben, míg a keret kék lesz.
Ha valami rendellenesség van, segítségünkre lehet a hibalehetőségek táblázata. Az is lehetséges, hogy a tv-készüléken a képélességet kell állítani.

Hibaelhárítás

I. A bekapcsolást jelző fény nem világít!
   a) Nincs bekapcsolva a számítógép, vagy valamelyik csatlakozó-kábel nem érintkezik.
   b) A számítógépben hibás a biztosíték. Ilyenkor vigyük el szakemberhez.
II. Nincs kép!
   a) A tv-kapcsoló vagy csatlakozó nem működik.
   b) Nem hangoltuk be megfelelően a tv-készüléket.
III. Kusza az ábra a tv-képernyőn!
   Ilyenkor a számítógép kikapcsolása után, a kazettát helyezzük be újra. Valószínűleg nem megfelelően volt betéve.
IV. Ha a kép nem színes, valószínűleg helytelenül hangoltuk a tv-t.
V. Ha van kép, de nincs hang, vagy a hangerőt, vagy a hangolást próbáljuk állítani.

Figyelem! Vannak olyan típusú tv-készülékek, melyek nem képesek a teljes PLUS/4 képernyő-kijelzésre, a két szélső oszlopot levágják. Ilyen esetben másik készülék használatát javasoljuk.
Ha ez nem lehetséges, a probléma megoldható a következő módon: nyomjuk meg először az ESCP billentyűt, majd írjuk be az R betűt. Ez a parancs kisebbíti a képméretet, így a teljes képmező megjeleníthető a képernyőn. Ezt a műveletet minden bekapcsoláskor vagy RESET után meg kell ismételni.

Perifériák

A perifériák olyan kiegészítő egységek, amelyek a PLUS/4-hez csatlakoztatva kiterjesztik a gép alkalmazhatóságát. Ilyenek: adattárolás, másolás, nyomtatás, lemezen vagy mágnesszalagon tárolt programok használata és a telekommunikáció által szolgáltatott információk felhasználása.

A programok megőrzéséhez és betöltéséhez adattároló berendezésre van szükség. Adatokat rögzíthetünk hajlékony mágneslemezen vagy kazettán. Ha lemezt használunk, lemez meghajtóegységre is szükségünk van. Amennyiben mód van rá, érdemes beszerezni, mert így a programtárolás gyors és megbízható lesz. A PLUS/4-hez használható típusok: Commodore 1541 és Commodore 1542.
Kazettás egységként a Commodore 1531 Datacasette típusú magnetofon a legmegfelelőbb.
Ha szövegszerkesztő programot vagy grafikus programcsomagot használunk, a képernyőn megjelenő információk nyomtatóval átvihetők papírra is. Többféle olyan nyomtató kapható, amely illeszthető a PLUS/4-hez: MCS–801, MPS–802, MPS–803 és DPS–1101. Ez utóbbi betűminősége levélíráshoz is megfelelő. A különböző nyomtatók különböző típusú nyomtatott képet adnak. Vásárláskor a számunkra legmegfelelőbb típust válasszuk ki! Megjegyzés: elképzelhető, hogy a tv-készülék nem ad olyan tiszta képet, mint az szükséges lenne. Ilyenkor a tv-t a video-bemenetről – ha van rajta ilyen – monitorként használhatjuk, és így tisztább és élesebb képet kaphatunk.

2. FEJEZET

A BILLENTYŰZET ÉS A KÉPERNYŐ HASZNÁLATA

A billentyűzetről általában
A speciális billentyűzet
Grafikus billentyűk
Programbillentyűk (F1...F8)
A HELP használata

A billentyűzetről általában

A számítógép billentyűzetén – a klaviatúrán – látható legtöbb nyomógomb ugyanolyan, mint egy közönséges írógépen, de itt valamennyit több feladatra is használhatjuk. Ebben a fejezetben a billentyűk használatával ismerkedünk meg. Bemutatjuk a speciális felhasználásukat, és azt is, hogyan írhatjuk be az előlapokon – homlokélen található grafikus jeleket. Természetesen hasznos és ajánlatos mindezt működés közben, azonnal kipróbálni.

A speciális billentyűzet

RETURN

Ez a leggyakrabban használt billentyű. Minden parancs, ill. utasítássor végén le kell nyomni! A beírt üzenetek csak ezután kerülnek be a számítógépbe.

SHIFT

Két, teljesen egyenértékű SHIFT gomb van a billentyűzeten azért, hogy a beírás minél kényelmesebb legyen. Feladata ugyanaz, mint az írógép váltógombjának: lehetővé teszi, hogy nagy- és kisbetűket, ill. itt grafikus jeleket is írjunk a programjainkba. A bal oldali SHIFT fölött elhelyezett SHIFT-LOCK gomb a SHIFT állapot rögzítésére való. Számítógépünk a bekapcsolás után automatikusan úgynevezett „nagybetű/grafika" üzemmódba kerül. Minden beírt betű nagybetűként jelenik meg a képernyőn. A SHIFT billentyűt benyomva válthatunk át a betű-gombok homlokélének jobb oldalán látható grafikus jelre. (A bal oldali grafikus jel használata a C= billentyűvel lehetséges.) A „nagybetű/grafika" üzemmódból a „nagybetű/kisbetű" üzemmódba – és vissza is – a SHIFT és a C= gombok egyidejű lenyomásával térhetünk át. Így a gép a beírt karaktereket kisbetűként jeleníti meg a képernyőn. Ebben az esetben nagybetűt a SHIFT és a betűt jelző gomb együttes lenyomásával írhatunk.

RUN/STOP

Programfutás közben használva, megállíthatjuk az éppen futó programot, ezzel a vezérlést, pl. újraindítást a billentyűzetnek adjuk át.
A SHIFT-tel együtt használva a RUN/STOP billentyűt, a meghajtóegységben

levő lemez első programját tölthetjük be. Beolvasás után a program automatikusan elindul.

Kurzor-mozgató billentyűk

Működés közben a képernyőn egy villogó jel, a kurzor látható, amely azt a pozíciót jelzi, ahol a következő karakter megjelenik. Könnyen és gyorsan mozgatható mind a négy irányban, a megfelelő nyíllal jelzett billentyű lenyomásával. Ezek a billentyűzet jobb alsó részén találhatók. Amíg valamelyik nyíl lenyomott helyzetben van, a jel folyamatosan halad a képernyőn, átugorja a már beírt karaktereket anélkül, hogy bármilyen hatással lenne azokra.

INST/DEL (beszúrás, törlés)

Ezzel a billentyűvel törölhetünk és beszúrhatunk karaktereket abba a sorba, ahová a kurzort írtuk. Lenyomására törlődik a kurzor bal oldalán levő karakter. Így a törlés után a többi karakter eggyel balra tolódik, ezzel eltűnik az üres karakter a sorból.

A SHIFT-tel együtt használva – INST funkció – üres karaktert állíthatunk elő az adott sorban, a kurzor bal oldalán és oda újra karaktert vihetünk be.

Az INST/DEL billentyű használatával sok időt takaríthatunk meg, ha módosítani akarunk a már beírt programokon. A javítás után ne felejtsük el a RETURN gombot benyomni!

CLR/HOME

A gomb lenyomásával a kurzort a képernyő bal fölső sarkába vihetjük, miközben a képernyő tartalma nem változik. A SHIFT-tel együtt használva törölhetjük a teljes képernyőt, a kurzor ekkor is a bal fölső sarokban marad.

CONTROL

A SHIFT-hez hasonlóan, mindig egy másik billentyűvel együtt kell használni! Feladata:

a) A képernyőre írt szöveg színének kiválasztása a megfelelő „szín" és CONTROL billentyű együttes lenyomásával.
b) A képernyőre vagy nyomtatóra való kiíratást, ill. listáztatást állíthatjuk meg a CONTROL és az "S" gombok lenyomásával. (Újraindítás bármelyik billentyűvel, kivétel a RUN/STOP!)
c) Az inverz kiíratáskor –RVS–, valamint a karakter villogtatáskor –FLASH– is szükséges a CONTROL, erre hamarosan visszatérünk.

Ezeken kívül néhány program használja a CONTROL-t saját, különle-

ges céljaira. (Megjegyzés: pl. a CONTROL és az E gomb lenyomása elviszi a képernyőről a kurzort, a CONTROL és £ visszahozza!)

C= (Commodor billentyű)

Ezt a billentyűt is más gombokkal együtt használjuk.

a) A nagy- és kisbetűk használatát teszi lehetővé, amint már láttuk a SHIFT-nél.
b) A karakterbillentyűk homlokélén látható bal oldali grafikus jeleket írhatjuk be, ha előbb lenyomjuk a C= gombot.
c) Ha a képernyőre írt szöveg színét akarjuk megváltoztatni, akkor a C= után nyomjuk meg a kívánt szín-billentyűt (fölső sor). Így a színbillentyűk homlokélén, az alsó sorban lévő színre válthatunk.
d) Egy program futását lassíthatjuk, ha folyamatosan lenyomva tartjuk a C= gombot. Felengedve, a futás az eredeti sebességgel folytatódik. (Ez az egyetlen eset, amikor a C= billentyű önmagában is dolgozik!)

A színbillentyűk

A képernyőre írható betűk, számok, grafikus jelek színét 16-féleképpen választhatjuk meg a PLUS/4-esen.

a) A színbillentyűk homlokélén, a felső sorban jelzett 8 szín valamelyikét a CONTROL gomb, majd utána a kívánt szín-gomb benyomásával kaphatjuk meg, pl. BLK: fekete.
b) A homlokéleken levő alsó 8 színhez a C=, majd a kiválasztott szín-billentyű lenyomásával juthatunk el, pl. ORNG: narancs.

Próbáljuk gyakorolni a színváltást. A képernyőre írt karakterek mindig az utoljára kiválasztott színűek lesznek. (A háttér színezését lásd később.)

RVS ON – RVS OFF

A REVERSE parancs rövidítése. Jelentése: inverz kiíratás. Ez azt jelenti, hogy ha pl. fekete betűkkel dolgozunk sárga háttérben, átválthatunk sárga betűkre és fekete háttérre. Az inverz képet úgy kapjuk meg, hogy a CONTROL és az RVS ON billentyűket nyomjuk meg. Mindaddig megmarad egy adott színösszeállítás a képernyőn, amíg a CONTROL és az RVS OFF billentyűkkel vissza nem állítjuk az eredeti színeket. Megjegyzés: programban is kiadhatjuk ezt a parancsot.

FLASH ON – FLASH OFF (villogtatás)

A képernyőn megjelenített karaktereket folyamatosan villogtatjuk a CONTROL és a FLASH ON parancsot tartalmazó billentyűk együttes lenyomásával. A CONTROL és a FLASH OFF használatával pedig megszüntethetjük a villogást.

Grafikus billentyűk

Amint már korábban említettük, bekapcsoláskor a számítógép „nagybetű/grafika" üzemmódba kerül. Ekkor használhatjuk a billentyűk homlokélén látható, több mint hatvan féle grafikus jelet.

A jobb oldaliakat a SHIFT és a megfelelő gomb, a bal oldaliakat pedig a C= és a megfelelő gomb lenyomásával tudjuk megjeleníteni.

A grafikus jeleket egymás fölé, mellé is írhatjuk, így képeket, ábrákat, táblázatokat rajzoltathatunk a képernyőre.

Természetesen egyéb grafikus megoldásokra is van módunk a PLUS/4-essel, ezeket részletesen tárgyaljuk a 7. fejezetben.

ESC (ESCAPE)

Ez a billentyű alkalmas számos képernyőszerkesztési feladatra, pl. „képernyőablak" készítéshez. Ezeknek az ablakoknak a helyét és méretét a kezelő állíthatja be a képernyőn. Így kijelölhetünk olyan munkaterületet, ahol dolgozhatunk anélkül, hogy a képernyő egyéb részein látható kép megváltozna. Az ESC billentyűt mindig valamely más karakter billentyűvel együtt használjuk. Először az ESC billentyűt, és utána a másikat!

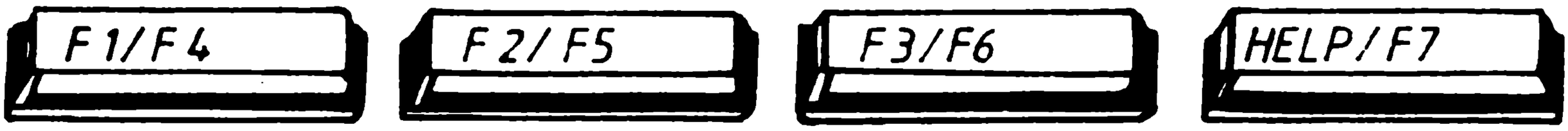
A: automatikus beszúrás
B: A képernyő-ablak jobb alsó sarkának beállítása, ott lesz ahová a kurzort elmozgattuk.
C: az automatikus beszúrás törlése
D: egy sor törlése
I: egy sor utólagos beírása
J: ugrás egy sor elejére
K: ugrás egy sor végére
L: képernyőgörgetés indítása
M: képernyő görgetés leállítása
N: az „ablak" törlése, visszatérés normál képernyőre
O: beszúrás, inverz kép és villogtatás leállítása
P: törlés egy sor elejétől a kurzorjelig
Q: törlés a kurzor helyétől a sor végéig
R: a képernyő méretének csökkentése
T: a képernyő-ablak bal felső sarkának kijelölése
V: képernyőgörgetés fölfelé
W: képernyőgörgetés lefelé
X: a kijelölt ESC feladat leállítása

Megjegyzés: ha képernyő-ablakot használunk, a B, D, I, J, K, P, Q, T, V, W billentyűkkel végezhető műveletek csak az ablak területére vonatkoznak.

Speciális jelek

Néhány olyan különleges jel is van a billentyűzeten, amely sem az írógépeken, sem néhány más számítógépen nem látható – ilyenek az angol font (£), a π, a kisebb-nagyobb jel (< >) stb. Ezek a karakterek jól használhatók bizonyos speciális programok elkészítésénél.

Programbillentyűk (F1...F8)



A billentyűzet fölött, jól elkülönítve a többi nyomógombtól látható ez a négy billentyű. Különböző, a programozás során gyakran ismétlődő feladatokhoz használhatók.

Az egyes billentyűk funkcióit a képernyőre írathatjuk a KEY parancs beírásával, majd a RETURN lenyomásával. A képernyőn akkor a következőt láthatjuk:

```
KEY1, "SYS1525 : 3 - PLUS - 1"
KEY2, "DLOAD" + CHR$ (34)
KEY3, "DIRECTORY" + CHR$ (13)
KEY4, "SCNCLR" + CHR$ (13)
KEY5, "DSAVE" + CHR$ (34)
KEY6, "RUN" + CHR$ (13)
KEY7, "LIST" + CHR$ (13)
KEY8, "HELP" + CHR$ (13)
```

Ezután a következő módon használhatjuk az egyes billentyűket:

F/1 : elindul a beépített szoftver csomag, a képernyőn megjelenik a következő felirat:
SYS 1525: 3 – PLUS – 1
Ezután a RETURN billentyűre indul a program.
(Más helyen visszatérünk ezekre a programokra!)

F/2 : a képernyőn megjelenik a DLOAD" parancs. Ha idézőjel közé beírjuk a lemezről betöltendő program nevét, a gép betölti az adott programot. A név után természetesen RETURN is kell!

F/3 : a képernyőn megjelenik a DIRECTORY (tartalomjegyzék) fölírás, és a lemezen található programok jegyzéke.

F/4 : képernyő törlés (grafikus üzemmódban is)

F/5 : a képernyőn megjelenik a DSAVE" parancs. Ha megadjuk a tárban

levő program nevét, majd lenyomjuk a RETURN-t, a számítógép átmásolja a programot lemezre.

F/6: programfuttatás, hatása ugyanaz mint a RUN parancsnak.

F/7 : a programlistát kiírja a képernyőre.

HELP: lásd később.

Az egyes feladatokat a megfelelő billentyű lenyomásával érhetjük el, az F/4; F/5; F/6; F/7 billentyűket a SHIFT-tel együtt aktivizálhatjuk.

Az egyes parancsokat többször is kiadhatjuk, közvetlenül és BASIC nyelvű programból is. Tetszőlegesen sokszor használhatjuk azokat, ill. közvetlen üzemmódban változtathatjuk is a parancsokat.

A HELP használata

Tulajdonképpen az F/8 billentyűnek felel meg. Ha hiba kerül a programunkba, a PLUS/4 leáll és hibaüzenetet ad. (A hibaüzenetek kódjaival később foglalkozunk.) Ekkor a HELP (Segítség!) gomb lenyomásával a hibás sor kiíródik a képernyőre, úgy, hogy a hiba villogva jelenik meg. Pl.:
A képernyőn a következőt látjuk:

```
?SYNTAX ERROR IN LINE 10
```

Ezután lenyomjuk a HELP billentyűt, és a képernyőn megjelenik a hibás sor, villogtatva. Pl.:

```
10 PRONT "..."
```

3. FEJEZET

SZOFTVER-HASZNÁLAT

Bevezetés
Beépített szoftver
Modulok
Kazetták
Mágneslemez

Bevezetés

A PLUS/4-en alkalmazható szoftver-család gyorsan gyarapodik! Minden szoftver termék – programok és egyéb eljárások – a programozó munkáját hivatottak könnyebbé és hatékonyabbá tenni.

Ez a számítógép alkalmas előregyártott programtároló modulon (Cartridge), kazettán vagy mágneslemezen tárolt szoftver fogadására. A használónak csak annyi a dolga, hogy megfelelően behelyezze ezeket a rendszerbe. Ugyanígy mi is készíthetünk és tárolhatjuk saját programjainkat!

Beépített szoftver

Ezek a beépített programanyagok a megfelelő programbillentyűkkel aktivizálhatók. Így kiterjeszthetjük a PLUS/4 működését a következő területekre: szövegszerkesztés, adatbázis-kezelés, grafikus működés és számvitel. Ez a programcsomag a gép bekapcsolásakor felhasználásra kész állapotba kerül. A KEY parancsot beírva, majd a RETURN gombot benyomva kiírathatjuk a képernyőre, hogy milyen programok érhetők el az egyes funkció (F) billentyűkkel. Megjegyzés: a fenti négy programterület alkalmazásával egy külön kézikönyv foglalkozik.

Modulok

A Commodore cég a PLUS/4-en használható modulprogramok széles választékát kínálja: személyi célú, üzleti, oktató, valamint izgalmas játékprogramokat. A modult a következőképpen kell használni és betölteni.

1. Kapcsoljuk ki a számítógépet.
   FONTOS: MIELŐTT BEHELYEZZÜK VAGY KIVESSZÜK A MODULT, A SZÁMÍTÓGÉPET KI KELL KAPCSOLNI; HA EZT ELMULASZTJUK MIND A SZÁMÍTÓGÉP, MIND A MODUL MEGHIBÁSODHAT!
2. Tartsuk a modult a feliratos oldalával fölfelé, és határozott mozdulattal dugjuk be a gép hátoldalán levő modulcsatlakozóba (memory expension).

3. Most bekapcsolhatjuk a számítógépet.
4. Kezdjük el a programot a mellékelt utasítás szerint. A modul-program a bekapcsolás után azonnal indul, az F billentyűk használata nélkül.

Kazetták

A PLUS/4-en használható programok széles választéka kapható magnetofonkazettán. Ezek a kazetták ugyanolyanok, mint a zenei kazetták, amelyeket magnetofonon használunk.
A Commodore program kazetták a Datacasette magnetofonról tölthetők be. Saját programjainkat is tárolhatjuk mágnesszalagon. Ebben a fejezetben részletesen tárgyaljuk, hogyan kell a programokat szalagon megőrizni és betölteni. A kazettáról való programbetöltéskor az eljárás ugyanaz, akár kész programot, akár saját programunkat használjuk.

Ne felejtsük el, még kikapcsolt állapotban, a kazettás egységet a számítógéppel összekötni a CASETTE feliratú aljzaton keresztül!

1. Helyezzük a kazettát a készülékbe, és zárjuk a fedelét.
2. Tekerjük a szalagot az elejére a REWIND gomb lenyomásával.
3. Ezután írjuk be a LOAD parancsot, majd nyomjuk meg a RETURN gombot. A képernyőn megjelenik:

   PRESS PLAY ON TAPE

4. A számítógép üzenete azt jelenti: nyomjuk meg a magnetofonon a PLAY gombot. Ha ezt megtesszük, a képernyő törlődik, a betöltés elindul. Amikor a mágnesszalagon programot talál a gép, a képernyőn a következő üzenet jelenik meg:

   FOUND "programnév"

   Az idézőjelben a megtalált program neve olvasható.
5. A program betöltéséhez nyomjuk meg a C= billentyűt. Miután a program bekerült a tárba, a képernyő visszatér az alapállapotba, és megjelenik újra a READY üzenet. A program futtatásához a RUN parancsot kell beírnunk. (Utána természetesen a RETURN gomb!)

A betöltött programot ki is listáztathatjuk a LIST paranccsal, és ha BASIC nyelvű a program, akár változtathatunk is rajta.
Egy program betöltését a RUN/STOP billentyű lenyomásával állíthatjuk meg.
Ha a kazettán a következő programra akarunk továbbmenni, nyomjuk meg bármelyik billentyűt kivéve a C= és a RUN/STOP-ot.
Ha egy meghatározott programot akarunk a kazettáról leolvasni, a LOAD után, idézőjel közé írjuk be a program nevét!

A programok felvétele kazettára

Ha programot írunk, és meg szeretnénk őrizni kazettán, a következőképpen járjunk el:

1. Írjuk be:

   SAVE "programnév"

   Az idézőjel közé a program nevét írjuk be. Ez bármi lehet, de nem lehet hosszabb 16 karakternél!
2. A parancsot természetesen most is a RETURN-nel fejezzük be. Hatására a képernyőn a következő jelenik meg

   PRESS RECORD AND PLAY ON TAPE

3. Nyomjuk meg a RECORD és a PLAY gombokat a magnetofonon. A képernyő törlődik, és amikor a program felvétele befejeződött megjelenik a READY üzenet.

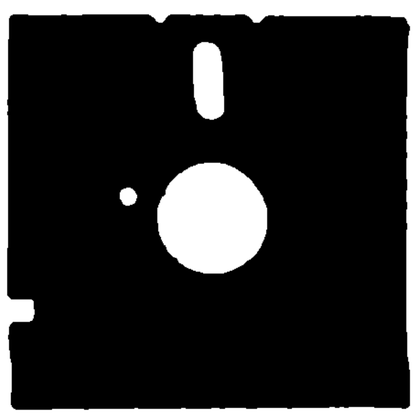
Megjegyzés: Amikor kazettára veszünk föl programot, mindig ellenőrizzük a kazettán futó szalag helyzetét. Ne vegyünk fel programot a kazetta legelejére, az úgynevezett ráfutószalagra, mivel az nem mágneses és így nem tud információkat tárolni.

A felvételt – hasonlóan a beolvasáshoz – a RUN/STOP billentyűvel bármikor megállíthatjuk.

Mágneslemez

A hajlékony mágneslemez – vagy angolul floppy disc – gyors és könnyen használható. Vigyázzunk, a meghajtóegységet és a lemezeket gondosan kell kezelni. A következőképpen használjuk:

1. Ellenőrizzük, hogy a meghajtóegység a géphez csatlakoztatva van-e, azután kapcsoljuk be.
2. Helyezzük a lemezt a meghajtóegységbe úgy, hogy a feliratos oldala legyen felfelé, és a lemeznek ez a része kerüljön be utoljára. A lemez oldalán egy kis bevágás található (lehet, hogy le van ragasztva). Ennek a bal oldalon kell lennie a lemez behelyezése során. A lemezt toljuk be egészen ütközésig!



3. A lemez betolása után zárjuk le a meghajtóegység védő fedelét.

4. Írjuk be: DLOAD "programnév"

   Az idézőjel közé most is az általunk választott nevet írjuk. A parancs beírása helyett az F2 billentyűt is lenyomhatjuk, ezután csak a programnevet és a második idézőjelet kell beírnunk.
5. A RETURN billentyűt lenyomva a lemez forogni kezd, és a képernyőn megjelenik a következő: (feltéve, hogy valóban található a lemezen ilyen nevű program!)

   SEARCHING FOR "programnév"
   LOADING
   READY
6. A program kész az indításra. Írjuk be a RUN parancsot, és nyomjuk meg a RETURN-t.

Ha a meghajtóegységen lévő piros fény villog a betöltés után is, valami hiba van! Írjuk be:

?DS$

Nyomjuk be a RETURN billentyűt. Ha a hibát nem tudjuk kideríteni, ne használjuk a lemezt!
Megjegyzés: a DLOAD "*" parancsra a lemezen található első program töltődik be. Olyan esetben célszerű használni. ha nem tudjuk a program nevét.

Üres lemez előkészítése (lemezformálás)

Minden üres lemezt elő kell készíteni a felhasználáshoz, a HEADER paranccsal. Vigyázat! Ne hajtsuk végre ezt a műveletet olyan lemezzel, amelyen valamit már tároltunk, hacsak nem akarjuk a lemez teljes tartalmát törölni. A HEADER parancs minden jelet letöröl a lemezről!
A HEADER parancs formája a következő:

HEADER "a lemez neve", U cím, lab, D szám

A lemez neve: bármely, 16 karakternél nem hosszabb nevet adhatunk a lemeznek.
U cím: itt jelöljük ki a periféria címét, amelyet használunk. Az első meghajtóegység címe: 8
lab: az I betű után bármilyen két betű, szám szerepelhet. Ez lesz a lemez azonosítója. Célszerű minden lemezt más és más azonosítóval jelölni.
D szám: a meghajtóegység száma. Ha kettős meghajtót használunk D0-t vagy D1-et írjunk. Egyes meghajtó esetén D0-t kell használni.

A HEADER parancs után, amint megnyomtuk a RETURN billentyűt, a képernyőn a következő kérdés jelenik meg:

ARE YOU SURE?

Jelentése: Bizonyos ön ebben? Mármint abban, hogy valóban törölni akarjuk a lemez tartalmát! A számítógép még egy lehetőséget ad, hogy döntésünket megváltoztassuk. Hátha mégis "megsajnáltuk" a lemezen lévő programot!
Ha a törlést elhatároztuk írjuk be az Y-t (YES) és nyomjuk be a RETURN-t. Amennyiben mégsem akarunk törölni, a kérdésre N-t (NO) válaszoljunk. Természetesen utána RETURN!
Nézzünk példát egy helyesen beírt HEADER parancsra:

```
HEADER "TORLES", U8, I1C, D0
```

Ha már ismerjük hogyan kell törölni, ill. formálni egy lemezt, akkor föl tudjuk venni saját programjainkat is mágneslemezre. (Később még részletesebben is megismerkedünk ezzel a paranccsal.)

Programok megőrzése lemezen

Ha egy programot később is akarunk használni, ne felejtsük el felvenni, és így elraktározni, mielőtt egy másik programot töltenénk be, vagy kikapcsolnánk a számítógépet. Különben a programunk elvész! Ha egy betöltött programon módosítottunk, akkor újra föl kell venni, ha az új változatot meg akarjuk őrizni!

Ha egy új programot az eredeti néven tárolunk, akkor az újjal felülírtuk a régit, így az megsemmisült. Ha mindkét változatot meg akarjuk őrizni, akkor az új változatnak felvételkor más nevet kell adnunk.

Egy program lemezre vitelét a következőképpen végezzük:

1. Írjuk be: DSAVE "a program neve"
   A program neve most is max. 16 karakter lehet!
2. Nyomjuk le a RETURN-t. A PLUS/4 a következő üzenetet írja ki

```
SAVING "programnév"
OK
READY
```

A felvétel kész, a programot átmásoltuk a mágneslemezre. Ha a lemezmeghajtón levő piros fény a DSAVE parancs után is villog, akkor valami hiba történt. Ekkor írjuk be a: ? DS$ parancsot, majd nyomjuk le a RETURN-t. Így – valószínűleg – megtudjuk a hiba okát.

A DIRECTORY parancs (tartalomjegyzék)

Amikor a programokat lemezre tároljuk, a számítógép listát készít a lemezen tárolt összes programról. Ez a lista tartalomjegyzékként megjeleníthető a képernyőn is, a következőképpen:

Írjuk le a DIRECTORY parancsot, majd nyomjuk meg a RETURN billentyűt. (Az F3 billentyűt lenyomva ugyanezt érjük el.) Ekkor a képernyőn megjelenik a lemezen levő programok katalógusa. Ha a DIRECTORY után, idézőjel közé írunk pl. egy C betűt, így: DIRECTORY "C*", akkor azokat a programokat írja ki a lemezről, amelyek C-vel kezdődnek.

4. FEJEZET

A PROGRAMOZÁS ELSŐ LÉPÉSEI

Bevezetés
Színek a billentyűzeten
Színes és inverz kiíratás
Néhány egyszerű program
Hibajavítás
Bevezetés a képernyő használatába
Képernyőre való íratás részletesebben
Képernyő ablakok használata

Bevezetés

Ennek a fejezetnek az a célja, hogy megismerkedjünk a PLUS/4 néhány jellemzőjével, és megtegyük az első lépéseket a programozási tanulmányainkban.

Színek a billentyűzeten

A képernyőn megjeleníthető karakterek színét változtathatjuk, hogy jobban olvashatók legyenek, vagy ügyes színkombinációval kiemeljünk bizonyos dolgokat.

3. Írjunk le néhány betűt, miután fölengedtük a CONTROL-t és a 6-ost. Igy minden leírt karakter zöld színben jelenik meg.

A CONTROL és az 1...8 billentyűk együttes használatával a színbillentyűk fölső részén található 8 színt választhatjuk.

CONTROL

Billentyű	Szín
1	fekete (Blk)
2	fehér (Wht)
3	vörös (Red)
4	encián (Cyn)
5	bíbor (Pur)
6	zöld (Grn)
7	kék (Blu)
8	sárga (Yel)

Most tartsuk lenyomva a C= billentyűt, és nyomjuk le így az 1...8 gombok valamelyikét. A kurzor, ill. a most leírt betűk a színbillentyűk alsó részén látható 8 szín valamelyikére változnak.

C =	
Billentyű	Szín
1	narancs (Orng)
2	barna (Brn)
3	zöldessárga (YlGrn)
4	rózsaszín (Pink)
5	kékeszöld (Blgrn)
6	világoskék (LBlu)
7	sötétkék (DBlu)
8	világoszöld (LGrn)

Színes és inverz kiíratás

A PLUS/4 16 különböző színben tud betűket, számokat és grafikus jeleket megjeleníteni. A képernyőn nemcsak a karakterek színét változtathatjuk, hanem a karekterek és a háttér színét is föl tudjuk cserélni. Ezt az inverz kijelzést így érhetjük el:

1. Töröljük a képernyőt. (SHIFT és CLR/HOME; vagy F4)
2. Tartsuk lenyomva a CONTROL billentyűt, és nyomjuk meg a RVS ON gombot. (9-es billentyű)
3. Engedjük föl ezeket a gombokat.
4. Használjuk most pl. a szóköz (SPACE) billentyűt! Ez a billentyűzet alján található hosszú billentyű! A szóköz billentyűt lenyomva, egy olyan színű csík jelenik meg a képernyőn, mint a betűk. Ha sokáig tartjuk így, a vonal a képernyő széléhez ér, és a következő sorban folytatódik tovább.
5. Engedjük most föl a szóköz billentyűt (RETURN nélkül).
6. Váltsunk karakterszínt (pl. CONTROL és valamelyik színbillentyű). A kurzor a kiválasztott színre vált.
7. A szóközt lenyomva, az előző csík folytatódik, de az újonnan választott színnel.
   A színváltást tetszőlegesen folytathatjuk tovább!
8. Térjünk vissza normál üzemmódba az inverz kiíratásról: CONTROL és RVS OFF billentyűk együttes lenyomása, majd RETURN

Nézzünk egy egyszerű feladatot

a) Írjunk valamilyen "üzenetet" inverzben! Pl. COMMODORE PLUS/4
   Az inverz betűk kiválóan alkalmasak elegáns fejlécek, címkék írására, bizonyos betűk kiemelésére.
   Szöveget a PRINT utasítás után, idézőjel közé kell írnunk. A COMMO-

DORE szót írjuk inverzben CONTROL és RVS ON) a PLUS/4-et pedig normál formában (CONTROL és RVS OFF).
Tehát írjuk a következőt:

PRINT "COMMODORE PLUS/4" (Utána RETURN!)

A Commodore leírása előtt ha inverz kiírásra váltunk, a kurzor helyett R betűt látunk, a PLUS/4 leírása előtt pedig – ha visszaváltunk RVS OFF-ra – egy grafikus jelet. Ez így jó, nem kell törődnünk vele, a kiírásnál már csak a kívánt szöveg jelenik meg. Ugye jól mutat!

b) Ugyanezt a kiíratást végezzük el úgy, hogy az inverz mód helyett villogtatjuk az első szót!
Emlékezzünk vissza a billentyűzet használatára!
(RVS helyett FLASH!) Ez talán még szebb!
Mindkét kiíratási módot használhatjuk majd programírás közben, utasításokon belül is!

Néhány egyszerű program

Első programunk nagyon rövid és egyszerű lesz, mindössze két sorból, két utasításból áll. Ne felejtsük el a programsorok elejéről a számokat! A számítógép ezek alapján tudja, hogy milyen sorrendben hajtsa végre az utasításainkat. Az egyes sorok végén a RETURN parancsot sem hagyhatjuk ki, ezt persze a billentyűzetről visszük be!

```
10 PRINT "PLUS/4"
```

Ez a programsor azt közli a géppel, hogy írja ki a képernyőre az idézőjel közé írt szöveget!

```
20 GOTO 10
```

Ez a sor pedig arra utasítja a gépet, hogy menjen vissza a 10-es sorszámú utasításra, és azt hajtsa ismét végre. Ha ezt a két utasítássort beírtuk, programunk – vagy inkább programocskánk! – bent lesz a számítógép tárjában. Erről akár meg is győződhetünk a LIST és RETURN parancsokkal.

A program végrehajtására a RUN parancs beírásával (utána RETURN!) utasíthatjuk a számítógépünket. Próbáljuk ki! Miért írja ki ilyen sokszor a képernyőre a PLUS/4-et? Mert a GOTO utasításra mindig visszamegy a 10-es programsorhoz, amely szerint ki kell írnia a képernyőre. Ezt az ismétlést huroknak, vagy végtelen ciklusnak nevezzük. A RUN/STOP gomb lenyomásával megállíthatjuk a programfutást.

Új feladat! Ha az előző programot ki akarjuk törölni a tárból, írjuk be a NEW parancsot, majd nyomjuk le a RETURN billentyűt. A számítógép ekkor a READY kiírásával válaszol jelezve, hogy készen áll az új programra. Írjuk be:

```
10 PRINT "COMMODORE"
20 COLOR 0, 7
RUN
```

Megfigyelhetjük, hogy a kiíratás most nem ismétlődik, hiszen nem írtuk be a GOTO utasítást, viszont a képernyőn a háttér színe kékre változott. Ezt a COLOR utasítással értük el: a 0 jelöli a hátteret, a 7 pedig a kék színt. Próbáljuk ki más számokkal is, de csak 1 és 16 között!

Hibajavítás

Ha beírás közben vétettünk hibát, a javításra több lehetőségünk van.
1. Bármikor újraírhatunk egy programsort, a PLUS/4 automatikusan kicseréli a régi sort az újra, miután lenyomtuk a RETURN billentyűt. Ilyenkor a régi sor is ott látható még a képernyőn, de a gép már nem veszi azt figyelembe. Ha két programsor szerepel ugyanazzal a sorszámmal, a gép csak az utoljára beírtat hajtja végre.
Pl.: a következő hibát a 10-es sor újraírásával javíthatjuk.

```
10 COLOR   0,3
20 PRINT "HATTER"
10 COLOR 0,3
```

Ellenőrzésképpen ha programlistát kérünk (LIST és RETURN, vagy F7), a helyes programsorok jelennek meg, a helyes sorrendben

```
10 COLOR 0,3
20 PRINT "HATTER"
```

Különben az új, módosított sornak hasonlítani sem kell az eredetire
A hibás 10-es sort így is átírhatjuk:

```
10 PRINT "GYAKORLAS"
```

2. Nem kívánt sort törölhetünk a programunkról, ha a sorszámát beírjuk és megnyomjuk a RETURN billentyűt. A számítógép ezután figyelmen kívül hagyja ezt a sort, még akkor is, ha az még a képernyőn látható. Erről a LIST paranccsal meggyőződhetünk.
3. Programsor javítása felülírással.

A kurzormozgató billentyűk használatával vigyük a kurzort a javítandó karakterre, és írjuk be az új karaktert. Természetesen, így több karaktert is újraírhatunk. Az eljárás végén nyomjuk meg a RETURN gombot, nem szükséges a kurzort végig vinni az egész programsoron. A gép "emlékezni" fog az egész sorra akkor is, ha felülírás után a sor közepén nyomjuk le a RETURN-t.
Próbáljuk ki ezt a javítási módot valamelyik programsorunkkal!
Fontos megjegyzés: Javításkor a kurzort elmozgathatjuk akármelyik programsorba, így a PRINT utasításra is. Azonban ha írunk egy új programot, és a PRINT "... után akarjuk a kurzort mozgatni, a rendszer meglepő dolgot csinál. Pl. a "kurzor jobbra" billentyű lenyomására a képernyőn megjelenik egy inverz J betű. Ha futtatjuk ezt a sort, az inverz J nem íródik ki, mert a gép ebben az üzemmódban minden beírt információt nyomtatásnak értelmez, és csak akkor hajtja végre – pl. a kurzor mozgatást – ha futtatjuk a programot.

4. Üres helyeket hozhatunk létre egy olyan sorban, ahová a kurzort elvittük. Az így kialakított helyre beírhatjuk a hiányzó karaktereket. A SHIFT gombot tartsuk lenyomva, és közben nyomjuk meg az INST/DEL billentyűt annyiszor, ahány karaktert akarunk beszúrni. Pl.: a következő sorban a C PLUS/4 szót ki szeretnénk egészíteni COMMODORE PLUS/4-re

   ```
   10 PRINT "C PLUS/4"
   ```

   Vigyük a kurzort a C betű után és nyissunk 8 betűhelyet (SHIFT és INST/DEL). A kurzor helyben marad, így beírhatjuk a hiányzó karaktereket:

   ```
   10 PRINT "COMMODORE PLUS/4"
   ```

   Ne felejtkezzünk el a RETURN-ről!
5. Törölhetünk egy sorból karaktereket vagy szóközt a DEL billentyűvel. Ennek az lesz a hatása, hogy a kurzortól közvetlenül balra álló karaktert törli, vagy a szóközt megszünteti. A DEL gombot egymás után többször is használhatjuk, így akármennyi karaktert kivehetünk a sorból, és természetesen újakat írhatunk a helyükre.
   Végezzünk el egy olyan programmódosítást, amely a COMMODORE 16 feliratból COMMODORE PLUS/4-et csinál!

Feladat:
Most, hogy megtanultuk programjainkat javítani, módosítani, kísérleteztünk már egy kicsit a számítógéppel. Készítsük el egy olyan feladat programját, amelynek a leírása egy kicsit tovább tart, de elég egyszerű kezdőknek is. Írassuk ki a képernyőre a következő szöveget:

Különös dolog történt velem
A billentyűk tanulmányozása közben
Találtam egy ♥ -et

A programot úgy készítsük el, hogy minden sor más színű háttérben legyen, és a szív villogjon a képernyőn. Ha valakinek az eddigi ismerkedés után nem sikerül ez a megjelenítés, ajánlatos egy kicsit lefékezni az olvasás sebességét. Visszalapozva a megfelelő helyre megtalálható minden szükséges információ. Jó munkát!
Akinek sikerült a program "belövése" és van még kedve játszani, próbáljon változtatni a szövegen és a megjelenítésen.

Bevezetés a képernyő használatába

Szöveg megjelenítése a képernyőn

Írjuk le az alábbi rövid programot! Minden sor végén a RETURN billentyűt nyomjuk le!

NEW	a tár törlése
10 PRINT "◘";	SHIFT és az S gomb
20 GO TO 10	

Ha lefuttatjuk (RUN) a programot, a képernyő megtelik szívekkel. Ha már a teljes képernyőt szívek borítják, állítsuk le a futást a RUN/STOP billentyűvel. Így tanulmányozhatjuk a PLUS/4 képernyőjét, akár le is számlálhatjuk, hány karakter fér el összesen. Itt jegyezzük meg, hogy a pontosvessző az oka, hogy a szíveket szorosan egymás mellé rajzolja a gép! Most tanuljunk valami újat! Számítógépünket nemcsak a GOTO utasítással kényszeríthetjük valamilyen tevékenység ismétlésére. A FOR...NEXT szerkezet (ciklus szervezés) alkalmas arra, hogy a közéjük írt utasításokat ismételten végrehajtsa a gép. (Lásd részletesen a BASIC utasításoknál.)
Írjuk le a következő programot:

```
NEW
10 FOR I= 1 TO 40
30 PRINT "◘";
40 NEXT I
```

Futtatva a programot, egy sor megtelik szívekkel, összesen 40 db-bal. Láthatjuk, hogy a képernyőn egy sorban 40 pozíció van, azaz a képernyő 40 oszlopból áll.
Most írjuk meg a programunkat a pontosvessző nélkül, és 40 helyett csak 25-ször ismételtessük a kiíratást.

```
10 FOR X=1 TO 25
20 PRINT "○"
30 NEXT X
```

A RUN parancsra a képernyőn az első oszlop telik meg körökkel. Pontosabban csak egy pillanatra telik meg, mert a program végén, az első 3 sorból eltűnik a jel. A READY üzenet elfoglalja a képernyő alsó három sorát. Mindenesetre megfigyelhetjük, hogy a képernyőn 25 sorba írhatunk összesen. Mindez azt jelenti, hogy a PLUS/4 a képernyőre 25 sorba, soronként 40 oszlopba, azaz összesen $40 \times 25 = 1000$ helyre tud megjeleníteni betűket, számokat, grafikus és egyéb jeleket.

Megjegyzés: előfordulhat, hogy különösen hosszú programsort, szöveget gépelünk be. Ha a szöveg 40 darabnál több karakterből áll, a gép automatikusan áttér a következő sorra, és ott folytatja a kiírást. Tehát nem kell megijedni, akár egy utasítás vagy művelet kellős közepén is elfogyhat a sor, még csak elválasztójelet sem kell használni.

Arra viszont ügyeljünk, hogy egy programsor max. 80 karaktert tartalmazhat, tehát maximum két teljes sorból állhat! A számítógép akkor tekint egy programsort befejezettnek, ha lenyomtuk a RETURN gombot.

Tekintsük a következő kis programot:

```
10 PRINT "*"
20 GOTO 10
```

A 10-es utasítást úgy írjuk, hogy a * mindkét oldalán hagyjunk egy-egy szünetet! Futtatáskor látjuk, hogy a gép kiíratáskor is minden * előtt és után kihagy egy-egy pozíciót, azaz a "szünetet" is kinyomtatja. Így áttekinthetőbbé tehetjük a kiíratási formát.

Képernyőre való íratás részletesebben

Próbáljuk ki a következő programot

```
10 PRINT "A", "B"
20 PRINT "A"; "B"
RUN
```

Amint látjuk, a képernyőn ez jelenik meg

```
A          B
AB
```

Tehát míg a két programsor csaknem azonos, a képernyőn jelentős a különbség. A megoldás kulcsa az írásjelek használatában van!

Ha a PRINT lista elemeit vesszővel választjuk el, kiíráskor a képernyőn egymástól helyközökkel elkülönítve jelennek meg. Ha pontosvesszőt használunk a kiírás közvetlenül egymás mellé történik, balról jobbra.

Ahogy láttuk, a képernyőn 40 oszlopban írhatunk. Az egész képernyő

négy, egyenként 10 karakter széles mezőre (PRINT ZONE) oszlik. Tehát a vesszővel elválasztott lista elemei a képernyőn balról jobbra haladva. az egyes zónák bal oldalán jelennek meg.
Próbáljuk ki akár parancsban, utasításszám nélkül is a kiíratást:

```
PRINT "A", "BC", "DEF"
```

Az eredmény ez lesz:

```
A       BC        DEF
```

Ha négynél több elemet íratunk ilyen formában a képernyőre, a gép miután elfogyott egy sorban a négy zóna, automatikusan áttér a következő sorba, és ott folytatja a megjelenítést.

Ha a pontosvesszőt használjuk a szövegrészek elkülönítésére, a gép nem veszi figyelembe a zónákat, a karaktereket balról jobbra, egymás után írja ki. Pl.:

```
PRINT "A"; "B"; "C"; "D"
```

A megjelent kép ez lesz

```
ABCD
```

Ha az első kiíratandó karakterlánc (angolul: string) 10 betűhelynél hosszabb, és vesszővel van elválasztva a többitől, a gép automatikusan két zónát tart fenn.

Töröljük a képernyőt, írjunk be egy NEW parancsot és a következő utasításokat:

```
10 PRINT 1,2
20 PRINT 1;2
RUN
```

A képernyőn megjelenő kép

```
1               2
1  2
```

A számok kiíratása tehát két dologban különbözik a betűk és egyéb karakterek megjelenítésétől.

a) A számokat nem kell idézőjelbe tenni. (Ez nemcsak a számokra, hanem minden változóra is vonatkozik, ha azoknak az értékét akarjuk kiíratni.)
b) A számokat a gép úgy írja ki, hogy mindkét oldalukon kihagy egy-egy szóközt. Tehát még pontosvesszővel sem tudjuk azokat közvetlenül egymás mellé íratni. Ez a szóköz hely az előjel helye!

Képernyő törlés

1. SHIFT és CLR/HOME billentyűk lenyomása
2. F4 billentyű lenyomása.

3. SCNCLR parancs beírása. Programban is szerepelhet utasításként.
4. PRINT "■" paranccsal. Az idézőjelbe látható grafikus jelet a SHIFT és CLR/HOME billentyűkkel írhatjuk le. Szintén kiadható ez az utasítás is programban.

Képernyő ablakok használata

A képernyőn kialakítható ablakok lehetővé teszik, hogy a képernyő egy részét kijelölhessük munkaterületnek. Így minden információ, amit a képernyőre viszünk – parancsok, lista, grafika – az ablak keretein belül jelenik meg, és nem befolyásolja az ablakon kívüli képernyőterületeket. Az oda előzetesen bevitt információk – keret, fejléc, felirat stb. – a programfutás során is a képernyőn maradnak.

A billentyűzet használatánál már láttuk, hogyan alakíthatunk ki ablakot a képernyőn.

1. A kurzort mozgassuk arra a képernyőpozícióra, amelyet az ablak bal fölső szakaszának szánunk.
2. Adjuk ki az ESCAPE–T parancsot, azaz nyomjuk le az ESC billentyűt és utána a T betű gombját.
3. Most vigyük a kurzort arra a helyre, ahol a jobb alsó sarkot akarjuk kijelölni.
4. Nyomjuk meg az ESC billentyűt, majd a B-t.

Az ablak elkészült, próbáljuk ki!

Ekkor minden kiírás, az így meghatározott "box"-ba kerül a képernyőn. (Lásd még az ESC billentyű használatát.) Ha az ablakot meg akarjuk szüntetni, készer egymás után nyomjuk meg a CLEAR/HOME billentyűt. Az ablak ezzel "eltűnik", és a kurzor a bal fölső sarokba kerül.

5. FEJEZET

SZÁMOK ÉS SZÁMÍTÁSOK

Számok és műveletek
Műveletek számokkal
Változók
Numerikus függvények
A véletlen számot előállító függvény

Számok és műveletek

A PLUS/4 használható egyszerű számológépként is. (Kalkulátor üzemmód). Műveleti jelei az összeadás: +, kivonás: −, szorzás: *, osztás: /, hatványozás: ↑. Ezek megegyeznek az összes többi, forgalomban levő számítógép műveleti jeleivel, és csak így érti meg a kijelölt műveletet a gép aritmetikai egysége. Ha pl. X-et használunk a szorzásra, azt betűnek értelmezné. A műveleti jeleket és számokat parancsként (PRINT), tehát sorszám nélkül, közvetlen üzemmódban, vagy utasításként, tehát programban egyaránt használhatjuk. Mindkét esetben idézőjei nélkül kell beírni! Pl.:

a) PRINT 2*3+1

Utána a RETURN-t lenyomni! Ezt a továbbiakban nem hangsúlyozzuk, feltehetően mindenki megszokta már!

b) 10 PRINT 2*3+1
RUN

A BASIC nyelvben használatos műveleti és relációjelek a következők:

Matematikai műveleti jelek:

Összeadás: +
Kivonás: −
Osztás: /
Szorzás: *
Hatványozás: ↑

Relációjelek:

Nagyobb: >
Kisebb: <
Egyenlő: =
Nagyobb vagy egyenlő: = >
Kisebb vagy egyenlő: < =
Nem egyenlő: < > vagy > <

Figyelem: A PLUS/4 – hasonlóan a többi számítógéphez – nem fogad el tizedesvesszőt, helyette tizedespontot kell írnunk! Pl.: 109,401 helyett 109.401! Ha egy számon belül vesszőt írunk, azt a számítógép két, külön, vesszővel elválasztott számnak veszi.

Törtek, tizedestörtek

Ugyanazt a törtszámot többféleképpen is beírhatjuk:

Pl.: 0.5 vagy .5 vagy 1/2. Az utóbbinál a gép ténylegesen végrehajtja az osztást, így a végrehajtás is lassúbb, esetleg pontatlanabb lesz.

Kiíratáskor az eredményt mindig tört- vagy egész számként kapjuk. Pl.:

```
PRINT 139/493+5
5.28194726
```

Számításokban, ha a π (3.14159256...) számot használjuk, a π jelzésű gombot nyomjuk le. Pl.:

```
PRINT π /374
5.28194726 E−03
```

Mit is jelöl ebben a számban az E−03 rész? Ez egy tudományos, nem teljesen elemi matematikai jelölésmód. A számítógép a tízes számrendszerbeli (decimális) számokat, ha azok −999999999 és +999999999 közötti tartományba esnek, közönséges egész vagy törtszámként jeleníti meg. Ha egy szám kívül esik ezen a tartományon, vagy esetleg törtszámként tartalmaz kilenc számjegynél többet, akkor használja a gép az előbbi formát a kiírásra. Mi is írhatunk be ilyen alakú számokat, a számítógép természetesen így is végrehajtja a kijelölt műveleteket. A számoknak ez a normál alakja, az az előnye, hogy a hosszú számokat kevesebb számjeggyel ábrázolhatjuk. A normál alak mindig két részből áll. Az első rész egy 0 és 10 közötti számból, a második (az E-vel kezdődő) pedig azt mutatja, hogy a tizedespontot a gép hová helyezi el.

pl.: 198505478 = 1.98505478 E+8

Egynél kisebb szám esetén, a második rész negatív lesz, jelezve, hogy a tizedespontot jobbra kell elmozgatni.

Pl.: 0.00359 = 3.59 E−3

Műveletek számokkal

Ha bizonyos matematikai műveletet akarunk a gépünkkel elvégeztetni, és az eredményt szeretnénk a képernyőn megjeleníteni, akkor a feladatot a PRINT utasítás után írjuk be, idézőjel nélkül!

(A PRINT helyett elég, ha ?-et írunk!)

```
Pl.:  10 PRINT 5 * 3 , 5 + 3
      20 PRINT 8 / 2 , 3 ↑ 2
      RUN
```

Ahogy látható – remélhetőleg mindenki ezt látja – a képernyőre a következő négy szám kerül:

```
 15    8
  4    9
```

Ez az első eset, amikor a PRINT utasításra nem pontosan ugyanaz jelenik meg, mint amit beírtunk. Ugyanis a számítógép elvégezte az utasításban kijelölt műveleteket, és az eredményt írta ki, két zónában.

Ne felejtsük el, hogy most nem szabad idézőjelet használnunk!

Ha látni szeretnénk az eredmény mellett a műveletben szereplő számokat, jeleket is, akkor szövegként, azaz idézőjelek közé ezeket is beírhatjuk! Pl.:

```
10 PRINT "SZORZAS"
20 PRINT " 8 * 7 "
30 PRINT 8 * 7
RUN
SZORZAS
8 * 7
 56
```

Megfigyelhetjük, hogy a 10-es és 20-as utasítássor pontosan ugyanazt íratja ki, mint ami az idézőjelben van, a 30-as utasítás hatására marad egy szóköz az eredmény előtt az előjelnek.

Módosítsunk most a 20-as soron a kurzor mozgatásával, a "Hibajavítás"-ban megtanultak alapján.

```
20 PRINT " 8*7+1 = "; 8*7+1
RUN
SZORZÁS
8*7+1 =  57
 56
```

Tehát ha a feladatot és az eredményt is látni akarjuk, akkor egyszer idézőjelbe írjuk be a feladatot, egyszer pedig anélkül: PRINT "1+1 = "; 1+1

Röviden összefoglalva azt mondhatjuk, hogy tetszőleges számítást írhatunk utasításként programba, sorszámmal ellátva, vagy megkaphatjuk az eredményt azonnal, ha PRINT parancsba írjuk a feladatot. Az első a programüzemmód, az utóbbi pedig a direkt vagy közvetlen üzemmód. Mindkét esetben kiírathatjuk az eredménnyel együtt magát az elvégzendő műveletet is, idézőjelek közé, szövegként:

```
PRINT "2 harmadik hatványa egyenlő:"; 2↑3
2 harmadik hatványa egyenlő: 8
```

A műveletek sorrendje

Már eddig is láttunk ra peldát, hogy egy utasításon belül több számítás is elvégezhető

```
PRINT 200+50/5
210
PRINT (200+50)/5
 50
```

Talán van aki meglepődött, és nem ezeket az eredményeket várta! Pedig a számítógép számolóegysége ugyanolyan logikával számol, mint ahogy mindannyian azt az iskolában tanuljuk! A műveletek végrehajtása balról jobbra történik, de ezen belül bizonyos műveletek előnyt élveznek a többivel szemben. Azt a sorrendet, amely szerint a gép végrehajtja a műveleteket, végrehajtási vagy prioritási sorrendnek nevezzük. Ez a következő:

Először: a gép a negatív számokat keresi meg, és a belső számábrázolásában elvégzi az előjelváltást.

Másodszor: elvégzi a kijelölt hatványozást.

Harmadszor: végrehajtja a szorzásokat és az osztásokat, balról jobbra.

Negyedszer: az összeadásokat és a kivonásokat, balról jobbra.

Ezt a sorrendet zárójelek használatával módosíthatjuk. Először mindig a zárójel tartalmát számolja ki! Zárójeles kifejezéseket ismételten is zárójelbe tehetünk. Pl.:

```
PRINT 3* (2↑2+1)
```

Ajánlatos bizonyos esètben a negatív számokat az érthetőség kedvéért zárójelbe tenni. Pl.: ?45*(−5). Számítógépünk zárójel nélkül is érti a feladatot: ? 45* −5

Változók

A következő példa a számítógép egyik legfontosabb tulajdonságát szemlélteti: 36*(12+A/3). Ha egy matematikai kifejezésben számok mellett betű is szerepel, akkor a program azt változónak tekinti. A változó mindig egy értéket képvisel, amely vagy egy előzetes számítás eredménye, vagy pedig a programban kell megadni.

Pl.:
```
10 A=6
20 PRINT 36*(12+A/3)
RUN
 504
```

Programjainkban háromféle változót használhatunk:

a) Egész típusú változók (Integer), amelyeket %-jellel definiálunk, és csak egész szám lehet az értékük. Pl.: A%=17 ; X%=15 stb.

b) Lebegőpontos változók, amely azt jelenti, hogy a tizedespont a számban tetszőleges helyen lehet. A programban nem kell külön jelölést alkalmazni. A számítógép automatikusan a valós egész és törtszámokat ilyen típusú számként kezeli.
   Pl.: X = 23.5 ; A1 = 1.3E + 2 ; TK = 5 stb.
c) Szöveg típusú változó, vagy sztring-változó, amely tulajdonképpen egy tetszőleges karaktersorozat. Mindig idézőjelbe kell tenni, a jelölésre a $ jel szolgál.
   Pl.: X$ = "ABC" ; M$ = "COMMODORE" ; stb.

Ügyeljünk arra, hogy mindig a helyes változótípust használjuk. Ha például egy szót akarunk megfeleltetni egész típusú változónak, a számítógép hibaüzenetet ad.

A következő programmal gyakorolhatjuk, hogy milyen adatokat adhatunk egy változónak az adott kapcsolatban.

```
10 PRINT "IRJUNK BE EGY EGESZ SZAMOT"
20 INPUT X%
30 PRINT "BRAVO, A SZAMOT IGY OLVASOM" ; X%
40 GOTO 10
```

Az egyik újdonság a programunkban a 20-es utasítássor. Az INPUT utasítással különböző adatokat, pl. ebben a programban az X% változó értékét vihetjük be a gépbe a billentyűzetről. Az INPUT utasításra a képernyőn egy kérdőjelet látunk, és a számítógép addig várakozik, amíg beírjuk az adatot.

Próbáljuk meg futtatni a programot a következő értékekkel: először 10, azután 0.43, majd EGYOTOD!
(Ugye mindenki kitalálta, hogy az X% változó miért csak a 10 lehet?!)

Numerikus függvények

A PLUS/4 számítógép BASIC 3.5 nyelve tartalmaz olyan numerikus függvényeket, amelyek már a komolyabb zsebszámológépeken is megtalálhatók (szinusz, tangens stb.). Ezeket a függvényeket úgy kell használni, hogy a függvény neve után zárójelbe tesszük azt a számot vagy változót, amellyel az adott műveletet el akarjuk végezni. Pl.: SIN(X) ; SQR(25) stb. Egy korábbi példában már szerepelt a FOR...TO...NEXT szerkezet. Nézzük most a következő programot, ahol a függvény egy utasításon belül szerepel:

```
10 FOR X = 1 TO 100
20 PRINT X ; "NEGYZETGYOKE:" ; SQR(X)
30 NEXT X
```

A program ha lefuttatjuk kiírja 1-től 100-ig az egész számokat és a négyzetgyököket.

Egy későbbi részben me'gtaláljuk a használatos függvények teljes listáját. Ez lényegesen több függvényt tartalmaz, mint a számológépeké.

A véletlen számot előállító függvény

10 papírdarabra fölírunk egy-egy számot 1-től 10-ig, és betesszük egy kalapba. Ha kihúzunk egyet, a rajta olvasható szám egy véletlen szám. Tegyük vissza minden kihúzás után a számot, és az egészet ismételjük meg valahányszor. Jegyezzük föl a leolvasott számokat! Soha nem tudhatjuk, hogy mi lesz a következő, csak annyit, hogy 1 és 10 között lesz! Ez a véletlen számok elve, igen fontos a programozásban, mivel a véletlenszerűen bekövetkező eseményekre következtethetünk belőle.

Nézzük meg a PLUS/4 hogyan állít elő véletlen számokat, és mire használhatjuk ezeket.

A következő program öt véletlen számot ír ki a képernyőre:

```
10 FOR X=1 TO 5 : PRINT RND(X): NEXT X
```

Megjegyzés: látható, hogy egy programsorba több utasítást is írhatunk, ha kettősponttal elválasztjuk ezeket egymástól. A képernyőn megjelenő számokból egész számokat képezhetünk az INT függvény felhasználásával, és így könnyebben használhatók programban.
A programban azt a tartományt is megadhatjuk, ahonnan a gép kiválasztja a véletlenszerű számokat. Ennek általános alakja a következő:

```
INT(A * RND(1)) + B
```

Ahol B jelöli az alsó határt, tehát azt a számot, amely a legkisebb véletlen számot jelenti, az A pedig, hogy hány darab számból választhat a gép. Így ha pl. 10 és 25 közötti számokat akarunk kiíratni, akkor B = 10 és A = 16.
Próbáljuk ki a következő programot többször egymás után lefuttatni:

```
10 PRINT INT(5*RND(1)) +1
```

Mindig 1 és 5 közötti számot látunk a képernyőn!
Ha azt akarjuk, hogy egymás után ismételje a gép, ezt az eljárást, akkor a véletlen szám kiválasztását egy ciklusba kell írnunk:

```
10 FOR I=1 TO 5
20 PRINT INT (90*RND(1)) +1
30 NEXT I
```

Ez a program a lottószámok véletlenszerű kiválasztását helyettesítheti, 1 és 30 között. (Játsszuk meg, hátha véletlenül a nyertes 5 számot választja ki a PLUS/4!)

Meghatározható függvények

Nem minden számítógép rendelkezik azzal a tulajdonsággal, hogy a programba új függvényeket is definiálhatunk! Pedig ez a lehetőség nagyon hasznos lehet különböző matematikai számítások ismételt végrehajtásakor, és alkalmazása egyszerű a PLUS/4-esen. Ezzel a lehetőséggel tárterületet takaríthatunk meg, és a futási idő is rövidebb lesz. A könyv végén lévő BASIC ismerettár 10. részében található egy lista, azokról a matematikai függvényekről, amelyeket új függvények definiálására használhatunk. Ennek formája a következő:

```
DEF FNR(X)=...
```

```
Pl.: 10 DEF FNR (X) = 1 / COS(X)
```

A következő programsor egy olyan függvényt ad, amely véletlen számokat állít elő 1 és X között.

```
10 DEF FNR(X) = INT(X*RND(1)) +1
```

Egy érdekes alkalmazása a most tanultaknak a következő program. (Az egyes utasításokra később visszatérünk.)

```
10 DEF FNR(X) = INT(X*RND(1))+1
20 DO
30 COLOR 1, FNR(16), 5
40 LOOP
RUN
```

6. FEJEZET

ISMERKEDÉS A BASIC PROGRAMOZÁSI NYELVVEL

Bevezetés
Programozási módok
Beviteli/kiviteli utasítások (I/O utasítások)
Vezérlő utasítások és ciklusok
Feltételes utasítások
Szubrutinok
Megjegyzés (REM)

Bevezetés

Már az eddigiekben is láttunk néhány példát a BASIC 3.5 programozási nyelv használatára, a PLUS/4 képességeinek bemutatására. Nem biztos, hogy minden olvasó, minden egyes utasítást pontosan megértett! Ez a fejezet még mindig csak bevezetés, olyan gyakran használt BASIC utasításokra koncentrál, amelyek okvetlenül szükségesek a programok megírásához. Nyugodtan kísérletezhetünk ezekkel, a számítógépnek semmi baja nem történik egy hibás programtól, legfeljebb nem hajtja végre! Ahhoz, hogy valaki igazán megtanuljon programozni, még más szakkönyveket is végig kell böngésznie, és ne feledjük, hogy a tanulás legjobb módja maga a programozás.

Programozási módok

Kétféle módon dolgozhatunk a PLUS/4-en, közvetlen (vagy parancs) üzemmódban, és közvetett (vagy program) üzemmódban.

Közvetlen mód: amint a név is jelzi, a számítógép a RETURN benyomása után azonnal végrehajtja a beírt parancsot. Pl.: PRINT 2↑3
Ahogy már korábban láttuk, sorszámot nem írunk a parancs elé. Ezt a módot használjuk a számolások elvégzésére, és a programokkal kapcsolatos szerkesztések, tárolás, törlés, futtatás vezérlésére. Pl.: PRINT, LIST, SAVE, RUN stb.

Program mód: arra ad lehetőséget, hogy egy BASIC utasítást, amely egy utasítás-rendszernek, a programnak része, az adott helyen végrehajtsa a számítógép. A program minden utasításssora sorszámmal kezdődik, amely közli a géppel, hogy milyen sorrendben kezelje és hajtsa végre azokat. Ne felejtsük, hogy mindig le kell nyomni a RETURN-t minden kész programsor végén, hogy bevigyük azt a PLUS/4 tárjába. Ha viszont már tároltuk az egész programot, semmi sem történik addig, amíg a RUN parancsot ki nem adjuk. Ez közli a géppel, hogy hajtsa végre az egyes utasításokat a legkisebb sorszámú programsorral kezdve. A sorokat leggyakrabban tízesével szokás számozni, így szükség esetén könnyűszerrel beírhatunk új sorokat is. Mindazonáltal a PLUS/4 rendelkezik egy paranccsal: RENUMBER (újraszámozás), amely lehetővé teszi, hogy új sorok hozzáadásával a meglevő sorszámokat megváltoztassuk. Ez a lehetőség gyakran kényelmesebbé teszi a programozást.

Beviteli/kiviteli utasítások (I/0 utasítások)

Ezek az utasítások arra használhatók, hogy a programot futtató személy kommunikálni tudjon a számítógéppel. Ha olyan programmal dolgozunk, amelyben minden szükséges adat rendelkezésre áll, akkor az adatbeviteli utasításokat nem szükséges használni. Közvetlenül beírhatjuk ezeket a programba. Az ilyen program azonban csak ezekkel, a már meglevő adatokkal fut, újakat nem vihetünk be. Hasonló egy "kőbe vésett felirathoz". Sokkal rugalmasabb, hasznosabb, ha programunkat általánosan, változók felhasználásával készítjük el, és futtatáskor adunk értékeket ezeknek a változóknak. Az adatkiviteli utasításokat arra használja a számítógép, hogy a végeredményeket közölje az alkalmazóval. Ez létfontosságú, hiszen semmi értelme nem lenne a programnak, ha csak a gép központi tárában lenne elraktározva az eredmény, és mi sohasem ismerhetnénk meg.

Ebbe a csoportba tartoznak azok az utasítások és parancsok is, amelyek a központi számítógép és a perifériás egység (lemezegység, kazettásegység, nyomtató) között az információcserét biztosítják. Erre is láttunk már példát:pl.: LOAD, SAVE stb.
Most a legfontosabbakat tekintjük át, részletesen megtalálható valamennyi I/0 utasítás a könyv végén levő BASIC ismerettárban.

PRINT

Jelentése: írd ki, nyomtasd ki – mármint a képernyőre! A korábbi példákban már láttuk, hogy ez nagyon rugalmasan használható utasítás, amelyet parancsként is használunk. Üzenetek, grafikus jelek, változó értékek, számítási eredmények kiíratására alkalmas.

1. *Szövegkiírás*
   Ha bármilyen szöveget akarunk a programban közölni a képernyőn keresztül, a legegyszerűbb mód, ha a PRINT utasításban, szövegként (sztring), idézőjelek közé írjuk a szöveget.

```
10 PRINT "FIGYELEM, KEZDODIK A PROGRAMOZAS,"
20 PRINT "KOVESSUK AZ UTASITASOKAT!"
```

Bármilyen karakter állhat a PRINT utasítás után idézőjelben; szavak, betűk, számok, írásjelek stb. A PRINT utasítás után a karakterláncban bizonyos billentyűk a normál működésüktől eltérően dolgoznak. Ilyenek pl. a kurzormozgató vagy a színbillentyűk. Ahelyett, hogy elmozdulna a kurzor, vagy megváltozna a szín, amikor ezeket a gombokat megnyomjuk, megjelenik egy inverz karakter. Ettől ne ijedjünk meg, ezek a karakterek nem kerülnek kiírásra, sőt a RUN után a feladat is végrehajtódik, amit

beírtunk. A számítógép számára az inverz jelek is azt a funkciót jelentik, amelyet az adott billentyű egyébként kijelöl.
Próbáljuk ki!

2. *Számok és műveletek kiíratása*
A PRINT-utasítással kiírathatjuk annak a műveletnek a végeredményét, amelyet az utasításban elvégez a gép. (Lásd korábban, a Számok és műveleteknél!)

Pl.:
```
100 PRINT 58*15, 27, 1000*(4–3)
RUN
 870  27  1000
```

Érdekesebb, és egyben áttekinthetőbb lesz a kiíratás, ha a változókat, a bevitt adatokat, a műveleteket is megjelenítjük. Pl. a következő programban a 2 egység sugarú kör kerületét számoljuk ki.

```
10 R=2 : K=2*R*π
20 PRINT "HA A KOR SUGARA:" ; R
30 PRINT "AKKOR A KOR KERULETE:" ; K
```

Minden PRINT utasítás végrehajtása után a kurzor automatikusan a következő sor elejére ugrik. Ha pontosvesszőt teszünk a PRINT lista után, akkor a következő kiírás is ugyanabban a sorban folytatódik. Pl.:

```
10 PRINT "EZ A KET MONDATRESZ" ;
20 PRINT "UGYANABBAN A SORBAN LESZ"
```

INPUT

Jelentése: adatbevitel. Ezzel az utasítással tetszőleges adatokat – számok, szöveg – juttathatunk a programba. Sőt az INPUT utasítás után, idézőjelek között még üzenetet is kiírathatunk, azaz PRINT funkciót is elláthatunk. Ez a kiírt üzenet segítheti a használót, hogy pontosan milyen adatokat is vár a számítógép. Ha a program szöveges változóval dolgozik (pl. A$), akkor a beírt karaktereket mint szöveget értelmezi. Az INPUT utasítás csak programban adható ki, és a számítógép egy, a képernyőre írt ?-lel jelzi, hogy várja az adatot. Egy INPUT utasítással több adatot is bevihetünk, ebben az esetben vesszővel kell elválasztanunk egymástól a változókat.
Példák adatbevitelre:

a)
```
10 PRINT "MI A NEVED";
20 INPUT A$
30 PRINT "UDVOZOLLEK"; A$
RUN
```

b)
```
10 INPUT "IRJ BE EGY SZAMOT"; A
20 INPUT "MOST EGY MASIKAT" ; B
30 PRINT A; "*"; B; " = "; A*B
40 PRINT "ELLENORIZD A SZAMOLAST"
RUN
```

GET KEY

Csak programban, tehát utasításként fogadja el a PLUS/4. Alkalmazásával a billentyűzetről vihetünk be karaktert a programba. Az utasítást egy változó neve követi:

```
GETKEY A$
```

Ha a programfutás ehhez az utasításhoz ér, akkor addig várakozik, amíg le nem ütünk egy billentyűt. (De csak egyet!)

A lenyomott billentyűn levő karakter egyben az A$ változó értéke lesz. A billentyű használata után nem kell RETURN, a program azonnal folytatódik. Pl.:

```
10 PRINT "NYOMJ LE EGY BILLENTYUT, MERT ARRA VAROK!"
20 GETKEY A$
30 PRINT "UGYES VAGY,"; A$; "VOLT A BILLENTYU"
```

READ/DATA

Ezzel az utasítással a legkényelmesebben adhatunk a programváltozóknak értéket. A READ utasításban definiáljuk a változókat, amelyek numerikus és karakterváltozók is lehetnek. A READ után egy DATA utasításnak **kell** következni, amelyben felsoroljuk a READ-ben szereplő változók értékeit, ugyanolyan sorrendben! Ügyelnünk kell tehát arra, hogy a két utasításban ugyanolyan típusú változó, ill. adat legyen a megfelelő helyen.
Hibás adatbevitel pl. a következő:

```
10 READ A$, B
100 DATA 21, X
```

Ennél a programrészletnél a számítógép hibaüzenetet ad: TYPE MISMATCH ERROR, mégpedig ezért, mert az A$ változó értéke elképzelhető, hogy 21, de a B változóé csak szám lehet és nem az X karakter.

Helyes adatbevitelt alkalmaz a következő program:

```
10 READ A$, B, C$
20 PRINT A$; B; C$
30 DATA PISTI, 12, EVES
RUN
PISTI 12 EVES
```

Vezérlő utasítások és ciklusok

Igen gyakran van szükség arra, hogy a számítógép az egymás után beírt utasítások sorrendjétől eltérjen, vagy egyes utasításokat többször egymás után – ismételten – végrehajtson. Ezzel a lehetőséggel futási időt takaríthatunk meg, és összetettebb feladatokat oldhatunk meg. A vezérlésátadó utasítások közlik a számítógéppel ezeket az "ugrásokat". Úgynevezett feltétel nélküli ugró utasítás a már ismert GOTO, a számlálással vezérelt utasítás a FOR...TO...NEXT, és a megszállott struktúra-programozóknak a DO/LOOP utasítás.

GOTO

Az utasítás formája a következő: utasításszám GOTO utasításszám.
Tehát pl.:

```
10 PRINT "*"
20 GOTO 10
```

Amikor a GOTO utasításhoz ér a program, ekkor a GOTO utáni sorszámmal jelölt utasítássorra ugrik, és attól az utasítástól folytatja a végrehajtást. Ez az eljárás mindaddig ismétlődik, amíg meg nem állítjuk a RUN/STOP billentyűvel. Az ilyen "örökké" ismétlődő folyamatot végtelen ciklusnak is szokás nevezni. Az ilyen jellegű, feltétel nélküli ugró utasítást ritkán használjuk programban. Direkt üzemmódban, parancsként is kiadhatjuk, pl.: GOTO 100. Az ismétlődés általában valami feltételhez kapcsolódik. A feltételes vezérlésátadást valósítja meg a következő utasítás.

FOR/NEXT

A ciklusutasítás általános alakja:

```
FOR ciklusváltozó = kezdőérték TO végső érték STEP lépésköz.
:
Néhány BASIC utasítás
:
NEXT ciklusváltozó
:
```

A FOR utasítás a NEXT-tel együtt ismétlődő ciklust – hurkot – alkot. Azok az utasítások ismétlődnek, amelyek a FOR-t és a NEXT-et tartalmazó sorok közé esnek a programban. A FOR utasításban szereplő változó elvégzi a ciklusszámlálást. A változó értéke a beállított kezdőértéktől függ és minden egyes hurokban úgy változik, ahogy azt a STEP után megadjuk. Ha nem

írunk STEP-et, automatikusan egyesével növeli a program a ciklusváltozó értékét. Amikor a futás a NEXT-hez ér, akkor változik az érték, és a számítógép ekkor vizsgálja meg, hogy elértük-e már a megadott végső értéket. Ha igen, akkor a program a NEXT-et követő utasításson folytatódik, ha pedig még nem, akkor a FOR utáni programsorra ugrik vissza.

Nézzük a következő egyszerű ciklusszervezésű programot:

```
10 PRINT "SZAMOLJUNK EGYESEVEL TIZIG"
20 FOR J=1 TO 10
30 PRINT J
40 NEXT J
50 PRINT J-1; "-IG SZAMOLTUNK"
```

Látható, hogy nem kellett a STEP-et használni. Ha visszafelé számolunk, már nem hagyhatjuk ki!

```
10 PRINT "VISSZAFELE SZAMOLAS"
20 FOR K=10 TO 0 STEP-1
30 PRINT "MOST"; K; "AZ ALLAS"
40 NEXT K
```

DO/LOOP

Ez az utasításkombináció a cikluskialakítás másik lehetősége. Hatásos, és talán rugalmasabb, mint a FOR...NEXT utasítás. Itt csak néhány lehetséges alkalmazását említjük meg. A ciklus magja, az ismétlődő utasítások a DO és a LOOP között vannak a programban. Ha semmi mást nem adunk meg, akkor így egy végtelen ciklust állíthatunk elő. (Hasonlóan a GOTO-hoz.)

```
10 DO: PRINT I
20 I = Y+1
30 LOOP
```

Ezt a "végtelen sokáig" futó programot a RUN/STOP-pal állíthatjuk meg.

Feltételt is szabhatunk arra, hogy meddig ismétlődjön a ciklusmag, az UNTIL utasítással. A program akkor "száll ki" a ciklusból, amikor az UNTIL után beírt feltétel teljesül.

Pl.:

```
10 DO: INPUT "ELEGEDETT VAGY A PLUS/4-GYEL?"; A$
20 LOOP UNTIL A$ = "IGEN"
30 PRINT "KOSZONOM"
```

A WHILE használatával mindaddig ismétlődik a hurok, amíg az utánaírt feltétel teljesül. Tehát éppen fordítva, mint az UNTIL-nál.

A DO/LOOP utasítás egyéb alkalmazásaira a könyv végén, a BASIC ismerettárban visszatérünk.

Feltételes utasítások

Ez az utasítás a számítógép egyik leghasznosabb képességét használja ki. Így tudjuk programjainkat, bizonyos feltételek teljesülésekor, különböző irányba elágaztatni.

IF/THEN

Szerkezete a következő:

IF feltétel THEN vezérlés ELSE vezérlés

Hogyan működik ez az utasítás? Úgy, ahogy a gyermekkorunkban sokszor hallott, feltételhez kötött engedély: "Ha megírtad a leckét, akkor mehetsz játszani, különben még tanulni kell!"

Talán a példa túlzottan gyakorlatias, de ugyanígy dolgozik a programban az IF...THEN...ELSE szerkezet is!

```
10 INPUT "IRJ BE EGY POZITIV SZAMOT";X
20 IF X>0 THEN GOTO 50 ELSE 30
30 PRINT "NEM ERTED A FELADATOT?"
40 GOTO 10
50 PRINT "UGYES VAGY"
```

Megjegyzés:

a) THEN GOTO 50 helyett használható: THEN 50

b) Az ELSE 30 elhagyható, ugyanis ha az IF és a THEN közötti feltétel nem teljesül, a program automatikusan a következő sorszámú utasítástól megy tovább.

A BASIC ismerettárban erre az utasításra is visszatérünk.

Szubrutinok (alprogramok)

Ha egy programnak van olyan része, amelyet a programfutás során több helyen kell végrehajtani, akkor érdemes ezt a részletet szubrutinként megírni.

GOSUB/RETURN

Az utasítás formája: GOSUB sorszám

RETURN

A hívást a főprogramból a GOSUB sorszám utasítással végezzük, a visszatérés a főprogramba a RETURN utasítással lehetséges. Ez most **nem** a RETURN billentyűt jelenti, hanem be kell írni a programba, utasításként. Tulajdonképpen a szubrutin hívás hasonlít a GOTO utasításra, de lényeges különbség az, hogy most a program "megjegyzi" hogy hol volt a GOSUB elhelyezve, és a RETURN utasítás után innen folytatja a futást. Ugyanazt a szubrutint többször is használhatjuk egy programon belül.

Készítsünk egy egyszerű játékprogramot, amelyet akár szerencsejátéknak is nevezhetünk. A feladat a következő:
A gép találjon ki egy egész számot 1 és 100 között (véletlenszám). Kérjen tőlünk is egy számot ebben a tartományban. Mi akkor győzünk, ha párosság szempontjából ellentétes számot írtunk be, mint a gép. Azaz pl. ha a program páros számot írt ki, és mi is, akkor a gép nyert.

A következő programot tanulmányozzuk mielőtt lefuttatnánk:

```
10 PRINT "IRJ BE EGY EGESZ SZAMOT 1 ES 100 KOZOTT"
20 INPUT N%
30 IF N% < 1% GOSUB 100 : GOTO 20
40 IF N% > 100 GOSUB 100: GOTO 20
50 PRINT "JOL VAN! A JATSSZUNK MEG!"
60 GOTO 20
100 PRINT "EJNYE! JOL ISMERED A SZAMOKAT?"
110 RETURN
```

Vigyázzunk! Ha a programban úgy szerepel RETURN utasítás, hogy előzőleg nem volt GOSUB, a program végrehajtása megáll, és a PLUS/4 hibaüzenetet jelez: RETURN WITHOUT GOSUB ERROR.

Megjegyzés (REM)

Ezzel az utasítással üzenetet adhatunk a program alkalmazóinak. Semmiféle hatása nincs a program végrehajtására csak akkor jelenik meg a REM utáni üzenet, ha kilistáztatjuk a programot. Így ha magyarázatot akarunk fűzni a már elkészült programhoz, akkor érdemes használni.
Az előző programot kiegészíthetjük pl. a következő sorral:

```
5 REM EZ EGY EGYSZERU JATEK
```

Összefoglalva

Amint a bevezetésben megjegyeztük, ennek a fejezetnek nem lehetett az a célja, hogy a BASIC nyelvről teljes ismertetést adjon. Csak néhány alapvetően fontos információt és ötletet közöltünk. A teljes BASIC utasításkészlet megtalálható a könyv BASIC ismerettár című részében, példákkal is illusztrálva. Gyakoroljuk a programozás minden lehetséges változatát, akár önállóan, akár szakkönyvekből! A programozás hasonlít arra, mint amikor sózott földimogyorót eszünk; ha egyszer elkezded, nem tudod abbahagyni! Bár a kettőt nem érdemes egyszerre csinálni!

7. FEJEZET

A GRAFIKA ÉS A SZÍNEK HASZNÁLATA

Grafikus karakterek
Karaktermozgatás (animáció)
A színek kezelése
Finomfelbontású grafika
Pontok, vonalak, feliratok
Négyszögek, körök, sokszögek, festés
Többszínű grafika

Grafikus karakterek

Minden betűbillentyűn és néhány egyéb billentyűn is két különböző grafikus karakter található. Mint már említettük a korábbi fejezetekben, a grafikus jelek kiírásához lenyomjuk a SHIFT vagy a C= billentyűt, miközben a kiválasztott grafikus jel gombját is lenyomjuk. Használjuk először csak a jobb oldali jeleket: kártyaábrák, egy teli és egy üres karika, és összekötő vonalak.

Ha a PLUS/4 nagybetű/grafika üzemmódban van, tartsuk lenyomva a SHIFT billentyűt, és máris a képernyőre vihetjük a kívánt karaktert.

A következő néhány példa segítségünkre lehet abban, hogyan lehet grafikus jelekkel különböző ábrákat készíteni.

1. gyakorlat: *Nagy kör*

1. Nyomjuk le a SHIFT LOCK billentyűt.
2. Írjuk le az U, majd az I betűket. (Ne lepődjünk meg!)
3. RETURN
4. Írjuk be J, majd K.
5. Nyomjuk le a RETURN billentyűt.

2. gyakorlat: *Kígyó*

1. Nyomjuk le SHIFT LOCK billentyűt.
2. Írjuk be a következő karaktersorozatot: U I U I U I
3. Nyomjuk le a RETURN billentyűt
4. Írjuk be a következő karaktersorozatot: K J K J K J
5. Nyomjuk le RETURN

3. gyakorlat: *Görbe rajzolása*

1. Nyomjuk le a SHIFT LOCK billentyűt
2. Írjuk be a következő karaktersorozatot: E D C * F R
3. Nyomjuk le a RETURN billentyűt.

4. gyakorlat: *Két kereszt rajzolása*

1. Nyomjuk le a SHIFT LOCK billentyűt
2. Írjuk be a következőket: M szóköz N szóköz B
3. Nyomjuk le a RETURN billentyűt
4. Írjuk be a következőket: szóköz V, szóköz * + *
5. Nyomjuk le a RETURN billentyűt
6. Írjuk be a következőket: N szóköz M szóköz B
7. Nyomjuk le a RETURN billentyűt.

Bizonyára mindenki észrevette, hogy a SHIFT LOCK billentyűt elég volt egyszer, a gyakorlatok kezdetén megnyomni, végig ebben az állapotban marad. Ha most még egyszer megnyomjuk, akkor ismét alapállásba kerül. Ezek után föltehetjük magunknak a kérdést, hogy a számítógép miért nem jelzett SYNTAX ERROR hibaüzenetet a RETURN után? Ugyanis az egyes sorokba olyan karakterek kerültek, amelyeket sem parancsként, sem utasításként nem ért meg a PLUS/4. Azért van így, mert a RETURN-t a SHIFT-tel együtt használva a számítógép nem utasítássornak értelmezi a bevitt karaktereket, hanem egyszerűen továbbmegy a következő sorra.

Az eddigi gyakorlatokban még nem használtuk a bal oldali grafikus jeleket. Ezek is pontosan úgy működnek, csak a SHIFT helyett a C= billentyű hatására írhatók a képernyőre. Most nincs a SHIFT LOCK-nak megfelelő rögzítő billentyű, így végig nekünk kell lenyomva tartani. A bal oldali grafikus jelek között, különböző vonalak és szögek is találhatók, amelyeket ábrák és táblázatok készítésére lehet használni.

Pl. egy szót a következőképpen húzhatunk alá: A kurzort a szó alá visszük, és a C= és a T gombokat addig tartjuk lenyomott helyzetben, amíg végig nem ér a vonal.

5. gyakorlat: *Ék*

1. Lenyomva C= billentyűt írjuk be a következőket: T Y U
2. Nyomjuk le a CONTROL és a 9-es billentyűt? (C= fölengedve?!)
3. Nyomjuk le a C= billentyűt, majd a következő karaktersort: I O P @ szóköz.
4. Nyomjuk le a RETURN billentyűt

6. gyakorlat: *Ablak rajzolása*

1. Tartsuk lenyomva a C= billentyűt és nyomjuk le a következő karaktereket: A R S
2. Nyomjuk le a RETURN billentyűt
3. Nyomjuk le a Q-t
4. Engedjük fel a C= billentyűt
5. Nyomjuk le a SHIFT és + billentyűt
6. Nyomjuk le ismét a C= billentyűt (Végig lenyomva tartjuk!)
7. Nyomjuk meg a W, majd a RETURN billentyűt
8. Nyomjuk meg a Z E X, majd RETURN billentyűt.

A bemutatott példákban csak néhányat használtunk a grafikus jelek közül. Most, hogy már lehet valami elképzelésünk arról, hogy mi mindent lehet kihozni ezekből az előre megszerkesztett jelekből, próbálkozzunk önállóan is. Rajzoljunk a képernyőre különféle ábrát!

Karaktermozgatás

A mozikban vetített film valójában állóképek sorozata, ahol minden kép egy kicsit különbözik az előzőtől. Az egyes képeket nagyon rövid ideig látjuk, azután máris jön a következő kép. Így a jelenet mozogni látszik. A számítógép képernyőjén az ábrák mozgatása ugyanígy történik. Először rajzol a program egy képet, majd egy kissé megváltoztatja. A PLUS/4 elég gyorsan dolgozik ahhoz, hogy lehetővé tegye az ábrák egyenletes mozgását, de kizárólag programban! Ugyanis az információkat a billentyűzetről nem tudjuk elég gyorsan bevinni ahhoz, hogy a mozgás folyamatosnak tűnjön. A mozgókép létrehozásához másodpercenként legalább 30 kép szükséges, csak így "téveszti" meg a szemünket. Tehát a rajzoláshoz programot kell készíteni, várni a másodperc törtrészéig, majd új ábrát rajzolni. A képrajzoló programhoz a PRINT utasítást használhatjuk, és az egész képet grafikus jelekből alakítjuk ki, amelyen egy-két karakter megváltoztatásával keltjük a mozgás hatását.

A következő program egy labda pattogását szimulálja.
(A programokban a PRINT utasítás után a billentyűkön látható kulcsszavak természetesen az adott gomb lenyomását, és nem a kulcsszó beírását jelentik!)

1. NEW

```
10 PRINT "HOME majd SHIFT és Q"
20 FOR L=1 TO 100 : NEXT L
30 PRINT "HOME majd SHIFT és W"
40 FOR L=1 TO 100 : NEXT L
50 GOTO 10
RUN
```

A 10-es és a 20-as programsorok üres ciklust tartalmaznak. Ezek végrehajtása alatt semmi sem történik, csak a megjelenített kép áll egy rövid ideig.

Megjegyzés: Sokkal érdekesebb, látványosabb lesz a mozgó ábra, ha a képet több grafikus karakterből állítjuk elő, azután néhányat megváltoztatunk, míg a többi az eredeti helyén marad. Mindezt az alábbi program szemlélteti.

Fontos! Valahányszor a SHIFT vagy a C= billentyűkre utalunk, ezeket mindig a következő billentyűvel együtt, azonos pillanatban kell lenyomni!

2. NEW

```
10 PRINT "HOME SHIFT M SHIFT W SHIFT N"
20 PRINT "szóköz C=  +  szóköz"
30 PRINT "SHIFT N szóköz SHIFT M"
40 FOR I=1 TO 100 : NEXT I
50 PRINT "HOME szóköz SHIFT W szóköz"
60 PRINT "C= T C=  +  C= T"
70 PRINT "szóköz C= G C= G"
80 FOR I=1 TO 100 : NEXT I
90 GOTO 10
```

Az animáció eddigi két példájában az ábra csak a képernyő egy kis területén dolgozott. A következő lépés az, hogy a mozgó figura körbemenjen a képernyőn.
A PRINT TAB(N) "..." utasítás azt eredményezi, hogy a képernyőn az N-edik pozíción kezdődik a kiíratás.
A következő program egy kígyót mozgat a képernyőn.

3. NEW

```
10 FOR A=0 TO 30
20 PRINT "SHIFT CLR"
30 PRINT TAB (A) "SHIFT U SHIFT I SHIFT U SHIFT I"
40 PRINT TAB (A) "SHIFT K SHIFT J SHIFT K SHIFT J"
50 FOR L=1 TO L00 : NEXT L
60 PRINT "SHIFT CLR"
70 PRINT TAB (A+1) "SHIFT I SHIFT U SHIFT I SHIFT U"
80 PRINT TAB (A+1) "SHIFT J SHIFT K SHIFT J SHIFT K"
90 FOR L=1 TO 100 : NEXT L
100 NEXT A
```

Használjuk a SHIFT és a Q billentyűket egyszerre. Így egy labda formájú karakter jelenik meg a képernyőn, amelynek a mozgatásával látványos játékprogramokat készíthetünk. A mozgást úgy érjük el, hogy töröljük az ábrát az eredeti pozíciójáról, és egy szomszédos pozícióba helyezzük. Figyeljük meg a következő programot!

4. NEW

```
10 PRINT "SHIFT CLEAR"
20 PRINT "szóköz SHIFT Q kurzor balra" szóköz;
30 FOR L=1 TO 50 : NEXT L
40 GOTO 20
```

A képernyõn egy mozgó labdát látunk. A program futását a RUN/STOP billentyűvel állíthatjuk meg.

A színek kezelése

A képernyő egyes részein különféle színeket állíthatunk elő. A keret, a háttér, ill. a megjelenített karaktereknek is különböző színük lehet. A karakterek színeit más megtanultuk beállítani a billentyűzet tanulmányozása közben. A képernyő egyéb részeinek színét a COLOR utasítással állíthatjuk be, melyet közvetlen üzemmódban is (parancs) alkalmazhatunk.

Írjuk be: COLOR 4,3
Amint látjuk, a parancs végrehajtásakor a képernyő kerete pirosra változik. A 4-es szám a keretet, a 3-as pedig a piros színt jelöli. (Emlékeztető: a RED szín a 3-as billentyűn található!)

Próbáljuk ki: COLOR 0,7

A COLOR szó után az első szám a képernyő területét jelöli, amelynek a színét változtatjuk. A 0 a háttér, az 1 a karakterszín, a 4 a keret kódja. (2 és 3 lásd később). A COLOR utáni második szám az adott képernyőterület színét határozza meg. Az egyes színekhez tartozó számok megfelelnek a színbillentyűk számozásának.
1–8: fekete,... sárga
9–16: narancs,... világoszöld
A kiválasztott szín fényereje is változtatható a COLOR utasítással. A harmadik szám, – amely 0-tól 7-ig változhat –, határozza meg az árnyalatot 0 jelenti a leghalványabbat a 7 pedig a legerősebbet. Tehát a COLOR utasítás általános alakja a következő:

COLOR terület, szín, fényerő

A következő program a PLUS/4 teljes színskáláját bemutatja. (Éles kék képernyőn, az összes színt, minden árnyalatban.)

```
10 COLOR 0,7,7
20 FOR M=0 TO 7
30 FOR N=1 TO 2
40 FOR L=1 TO 16
50 PRINT "CONTROL RVS ON";
60 READ A
```

```
70 COLOR 1, A, M
80 PRINT "szóköz szóköz";
90 NEXT L
100 PRINT
110 RESTORE
120 NEXT N
130 NEXT M
140 COLOR 1    2.4
150 DATA 7,14,4,13.6.16.11,8,10,9,3,12,5,15,2,1
RUN
```

Megjegyzés: még egyszer hangsúlyozzuk, hogy a COLOR utasítás – hasonlóan a fejezetben előforduló többihez – parancsként is ugyanúgy használható. Csak a programozón múlik, hogy direkt vagy program üzemmódban használja-e.

Finomfelbontású grafika

Már korábban említettük, hogy a PLUS/4 képernyője összesen 1000 karakterpozíciót tartalmaz. (25 sor, 40 oszlop) Minden karakter különálló 8*8 pontból alakul ki. Így a teljes képernyő tulajdonképpen 64 000 pontból áll, amelyek mind külön-külön is megjeleníthetők! Gondoljuk csak el, milyen pontos, részletes és változatos képeket lehet így a képernyőre varázsolni! Már a billentyűzetről bevihető grafikus karaktereket használva is látványos rajzokat, játékprogramokat készíthetünk, de a finomfelbontású grafika segítségével sokkal aprólékosabb, finomabb lesz az ábra. Kiadhatunk olyan utasításokat, amelyekkel pontokat, vonalakat, köröket és más görbéket lehet rajzolni és törölni.

Van azonban ennek a rajzolási módnak egy korlátja is! Minden 8*8-as mezőben csak két színt használhatunk, egyet a háttér, a másikat pedig a pontok színezésére. A másik grafikus üzemmód, a többszínű grafika az egyes karakterpozíciókban 4 különböző színt használ, de ez a grafikai pontosság rovására megy. (Lásd később.)
Az itt következő program, a PLUS/4 finom felbontású grafikáját mutatja be, a GRAPHIC utasításon keresztül.

```
10 COLOR 0,1
20 GRAPHIC 1,1
30 FOR L=2 TO 16
40 COLOR 1, L, 7
50 DRAW 1,0, L*12 TO 319, L*12
```

```
60 DRAW 1,L*18,0 TO L*18,199
70 NEXT L
80 FOR I=1 TO 5000 : NEXT I
90 COLOR 1,2,7
100 GRAPHIC 0
```

Ugye látványos a képernyő? Tapasztaljuk azonban, hogy a metszéspontok környezetében a színek indokolatlanul megváltoznak. Ez nem szándékos, az oka az, hogy túl sok színű képpont van túl közel egymáshoz.

A képernyőn látható kép mellett több, eddig nem használt utasítással is találkozhatunk a fenti programban. Nézzük meg ezeket részletesen.

A finomfelbontású grafikára a GRAPHIC utasítással válthatunk át. A GRAPHIC 1,1 a teljes képernyőt használja, így egyetlen beírt parancs vagy utasítás sem jelenik meg a képernyőn. A GRAPHIC 2,1 azt eredményezi, hogy a PLUS/4 két külön tartományra osztja a képernyőt. Az alsó 5 sor megmarad normál képernyőként, utasítások, szöveg kiíratására, míg a fölötte levő rész használható az ábrák megjelenítésére.

A GRAPHIC 0 parancs – ez is kiadható programban – visszaállítja a képernyőt az alaphelyzetbe.
Általában a GRAPHIC utasítás formája a következő

GRAPHIC a,b

Az első szám 0 és 4 között változhat, ez a grafikus megjelenítés kódja. A számok hatása a következő:

0 : normál üzemmód, szövegkijelzés
1 : finomfelbontású grafikus üzemmód
2 : finomfelbontású grafika és szöveg
3 : több szín
4 : több szín és szöveg

A második számot nem kell okvetlenül megadni, a képernyő törlését jelenti

0 : a képernyő nem törlődik
1 : képernyőtörlés

A képernyőt törölhetjük az SCNCLR paranccsal is, vagy az F4 billentyűvel.
Megjegyzés: A finomfelbontású grafika 10 k memóriakapacitást igényel a képernyő használatára. Ezt a BASIC programterületből veszi el. Ha már nem használunk grafikát adjuk ki a GRAPHIC CLR parancsot, ezzel visszakapjuk a tárnak ezt a területét is.

Pontok, vonalak, feliratok

Mi történik a következő parancs hatására?

GRAPHIC 2,1 : DRAW 1,0,0

Ha egyáltalán látunk valamit, akkor a bal fölső sarokban egy kis pontot.
A **DRAW** utasítással rajzolhatunk pontot vagy vonalat.
Pont rajzolására:

DRAW a,b,c,

Az első szám 0 vagy 1 lehet. Kijelöli, hogy a háttér színével rajzoljon, ha 0, és a kurzor színével ha 1. Az első eset tulajdonképpen törlésre használható. A második és harmadik szám a megrajzolt pont sor-oszlop helyzetét határozza meg. A

DRAW 1,17,20

kirajzol egy pontot a 17. sor 20. oszlopába. Ugyanezen pont törlése:

DRAW 0,17,20

Ezzel a paranccsal egyenes vonalat is húzhatunk két pont között, ha megadjuk mindkét pont helyzetét a képernyőn.

DRAW 1,10,10 TO 100,100

Ugye látható a vékony kis szakasz a 10. sor 10. helyétől a 100. sor 100. helyéig? A képernyőnek ez a használata hasonlít a matematikában a koordináta-rendszerhez. A PLUS/4 azonban a kezdőpontot, a 0,0 pontot a képernyő bal fölső sarkába helyezi. Figyeljünk erre a mozzanatra, meg kell szokni!
A DRAW utasítást használhatjuk olyan változatban is, hogy az első pont helyzetét nem írjuk be. Ilyenkor a vonalat a legutolsó kirajzolt ponttól indítja a számítógép, persze valahol kell lennie egy "első" pontnak. Ennek a helyzetét a
LOCATE a,b
utasítással adhatjuk meg. Tehát az előző szakaszt így is megrajzolhatjuk:
LOCATE 10,10 : DRAW 1 TO 100, 100
Ne felejtsük el, hogy az egyes pontok is törölhetők, ha a DRAW után 0-át írunk.
Ez a program egy komoly matematikai függvény, a szinusz függvény görbéjét rajzolja ki:

```
10 COLOR 0,1
20 COLOR 2,1
30 GRAPHIC 1,1
40 LOCATE 0,100
50 FOR X =1 TO 319
60 Y = INT (100+99* sin ( x /20))
70 DRAW 1 TO X,Y
80 NEXT X
90 GOTO 90
```

Látjuk, hogy a görbét a szomszédos pontok összekötésével rajzolta a program. A 90-es sor miatt csak a RUN/STOP-pal léphetünk ki a végtelen ciklusból.
Ha a 70-es sort átírjuk:

```
70 DRAW 1,X,Y
```

akkor a teljes görbét pontokból kapjuk meg.

CHAR

A képernyőn megjelenített ábrák könnyebben használhatók, ha elnevezzük őket. A CHAR utasítással szöveget is írhatunk a képernyő bizonyos helyére, a finomfelbontásban készült ábrák köré!

A CHAR után az első szám 1 vagy 0 lehet. Az 1-es az írást, a 0 a törlést jelenti. A következő két szám azt az oszlopot, ill. sort jelöli, ahol a szövegírás kezdődik.

Egészítsük ki az előző programot ezzel a két programsorral:

```
85 CHAR 1,0,0, "SIN FV"
86 CHAR 1,0,1, "KEPE"
```

Négyszögek, körök, sokszögek, festés

A DRAW utasítással sok-sok pont és vonal elhelyezésével különböző alakzatokat rajzolhatunk. Négyzetet kapunk a következő parancs hatására
DRAW 1,0,0 TO 100,0 TO 100,100 TO 0,100 TO 0,0

Tulajdonképpen négy pontot, a csúcsokat kötöttük össze négy vonallal. Ugyanígy tetszőleges sokszöget is rajzolhatunk.

BOX

Ez az utasítás a derékszögű idomok (pl. négyzet) rajzolását könnyíti meg. Elegendő a BOX után beírni a négyzet két szemben levő csúcsának koordinátáit.

```
BOX 1,0,0, 100, 100
```

Az első 1-es most is a rajzolást jelenti – ugyanúgy, mint a DRAW-nál, a 0,0 pedig a bal fölső csúcs helyzetét. A kirajzolt négyzetet forgatni is lehet a képernyőn, az utolsó koordináta után beírt szöggel (fokban megadva). Pl.:

```
BOX 1,0,0, 100,100, 10
```

Ekkorr 10 fokkal fordul el a négyzet a középpontja körül, az óramutató járásával megegyező irányban.
Amennyiben nemcsak a négyzet körvonalait akarjuk kirajzolni, hanem a belső pontokat is – satírozás –, még egy 1-est kell az utasítás végére írni. Vigyázat, a szöget még akkor is jelezni kell – legalább egy vesszővel – ha nem forgatjuk a négyzetet!

```
BOX 1,100,50,220,150,,1
```

A következő program egy négyzetet forgat el különböző szögekkel:

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 2,1
40 A=20*RND(1) +10
50 FOR L=0 TO 359 STEP A
60 BOX 1,100,30,220,130,L
70 NEXT L
80 FOR I=1 TO 2000: NEXT I
90 GOTO 10
```

CIRCLE

Kör rajzolása

A PLUS/4-es külön parancsával lehet köröket rajzoltatni. A BOX-hoz hasonlóan a kör helyét, méretét változtathatjuk, rajzolhatunk íveket, ellipszist, sőt ezeket elforgathatjuk.

Próbáljuk ki a következő parancsot. (Természetesen utasításként is használható.)

```
CIRCLE 1, 160,100,50
```

Ugye milyen egyszerű! A számok jelentése:

1: azt jelzi, hogy rajzolja a vonalat. Ha itt 0 áll, akkor töröl.
160: a középpont a 160. oszlopban lesz.
100: a középpont a 100. sorban lesz.
50: a kör sugara 50 egység

Ugyanezzel a paranccsal szabályos háromszögeket, sokszögeket is rajzoltathatunk a képernyőre.

```
CIRCLE 1,160,100,50, 50,,,,120
```

Amint látható, meg kellett még adni azt, hogy az alakzat szabályos tehát körbe írható, ezt jelzi a második 50, majd 4 db vesszővel elválasztva az a szög, amely közli a számítógéppel, hány fokos szögben forgassa el a középpont körül az egyik csúcsot, hogy a másikba menjen át.

Ha N oldalú szabályos sokszöget akarunk rajzolni, ismerni kell hogy a középpontból nézve az elforgatás szöge most 360/N fok. (Gondoljuk csak meg!)
A következő kis program bemutatja ezt a lehetőséget:

```
10 GRAPHIC 2,1
20 INPUT "HANY OLDALA LEGYEN A SOKSZOGNEK";A
30 IF A<2 OR A>100 THEN PRINT "UJ ADAT": GOTO 20
40 CIRCLE 1, 160,100,40,40,,,,360/A
50 GOTO 20
```

Rajzolhatunk teljes kör helyett körívet is.
A rajzon tanulmányozhatjuk, hol kezdi a képernyőn a rajzolást a számítógép.

A jobb alsó negyedkört a következő parancs jeleníti meg:

```
CIRCLE 1, 160, 100, 50, 42, 90, 180
```

Itt 42 az ív "magassága", 90 az ív kezdőpontja (a kezdeti elforgatás szöge) 180 pedig a végpontja.

Ellipszist úgy rajzolhatunk, hogy a kört rajzoló parancsban a sugár után, amelyik most az ellipszis nagytengelyét jelenti, még egy számot megadunk, amely a kistengelyét jelöli ki.
Pl.:

```
CIRCLE 1, 160,100, 80,40
```

Természetesen az ellipszist is elforgathatjuk a középpontja körül, az óramutató járásával megegyező irányban, adott szöggel:

```
CIRCLE 1, 160,100,100;40,,,30
```

Tehát a CIRCLE utasítás (vagy parancs) lehetséges változatai a következők:

CIRCLE 1, a,b,c,	: kör
CIRCLE 1, a,b,c,d,	: ellipszis
CIRCLE 1, a,b,c,d,e,f	: körív
CIRCLE 1, a,b,c,d,,,e	: ellipszis elforgatása
CIRCLE 1, a,b,c,d,,,,360/N	: szabályos N-szög

A következő programot érdemes tanulmányozni! NEW-val kezdjük!

```
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 1,1
40 A=20*RND(1)+10
50 FOR I=0 TO 359 STEP A
60 CIRCLE 1,160,100,80,40,,,I
70 NEXT I
80 FOR J=1 TO 2000 : NEXT J
90 GRAPHIC 0,1
```

Most nézzük meg, hogyan használjuk a CIRCLE utasítást olyan körök rajzolásához, amelyeknek adatait változókkal határoztuk meg.

```
NEW
10 COLOR 0,1
20 COLOR 1,2
30 GRAPHIC 2,1
40 FOR I=1 TO 4
50 Y=50
60 IF L=2 OR L=4 THEN Y=100
70 X=L*35+50
80 CIRCLE 1,X,Y,50
90 NEXT I
100 PRINT "PLUS/4:KOROK"
```

PAINT (festés)

Ha egy alakzatnak nemcsak a határvonalát akarjuk megrajzolni, hanem ugyanilyen pontokkal a belsejét is ki akarjuk tölteni – festeni! –, akkor használjuk a PAINT parancsot. Természetesen kiadhatjuk utasításként is! Ha a képernyőterület nem zárt alakzat, a festés a képernyő széléig tart. Ugyanakkor teljesen szabálytalan – de zárt – alakzatok is kiszínezhetők így! Az utolsó programban kirajzolt négy kör meglehetősen bonyolult síkidomot határoz meg. Ezt a síkidomot, a belsejéből kiindulva, a következő programmal tudjuk befesteni:

```
110 FOR L=0 TO 1
120 PAINT 1,120+35*L,75
130 NEXT L
```

Többszínű grafika

Amint már láttuk, a finomfelbontású grafikával minden egyes pontot külön-külön is vezérelhetünk, de a különböző színű pontok már nem kerülhetnek tetszőlegesen közel egymáshoz. Ezek a grafikai programok csak 1–2 színt használnak.

Ahhoz, hogy több, különböző színt használhassunk, egy speciális, közbülső grafikai üzemmódot, a többszínű grafikát kell alkalmazni. Így minden sorban a grafikus pontok kétszer olyan szélesek lesznek, azaz egy sorban 160 ponttal rendelkezhetünk. A sorok száma nem változik, viszont a sokféle színhatásnak az az ára, hogy a rajzok kevésbé lesznek részletesek, pontosak.

Már láttuk korábban, hogy a többszínű képernyő a GRAPHIC 3 paranccsal állítható be, ill. ha alul 5 sorba szöveget is akarunk íratni, a GRAPHIC 4 paranccsal. Ezenkívül a COLOR utasításnál nem használtuk még a 2-es és 3-as paramétereket. Ezzel újabb 1 vagy két színt használhatunk a karakterszínen kívül. Előnye, hogy ezek között a színek között nem lesz átfedés.

A következő két program a többszínű grafika két alkalmazását mutatja be.

a)
```
NEW
10 COLOR 0,1
20 GRAPHIC 4,1
30 FOR L=1 TO 4
40 Q=L : IF Q>3 THEN Q=Q-3
50 COLOR Q, L+1
60 Y=50
70 IF L=2 OR L=4 THEN Y=100
80 X=L*18+25
90 CIRCLE Q, X, Y, 25, 42
100 NEXT L
```

b)
```
NEW
10 COLOR 0,1
20 GRAPHIC 3,1
30 COLOR 3,1
40 TRAP 200 : REM HIBAKEZELES
50 DRAW 3, 10, 10 TO 10, 100
60 DRAW 3, 10, 55 TO 30, 55
70 DRAW 3, 30, 10 TO 30, 100
80 DRAW 3, 50, 10 TO 80, 100
90 DRAW 3, 65, 10 TO 65, 100
100 DRAW 3, 50, 100 TO 80, 100
```

```
110 FOR L=0 TO 7
120 COLOR 3,2, L
130 FOR M=1 TO 100 : NEXT M
140 NEXT L
150 COLOR 3,1
160 FOR M=1 TO 100 : NEXT M
170 GOTO 110
180 GRAPHIC 0 : COLOR 1, 2, 7
```

Láthatjuk, hogy a COLOR 3, utasítás egészen különleges tulajdonságokkal rendelkezik. A képernyő bármely területére így rajzolt színt megváltoztathatjuk. Ha pl. a színt a COLOR 3,5-tel bíbor színűre állítottuk be, a COLOR 3,6 hatására – amikor ide ér a programfutás – minden bíbor tartomány zöldre változik.

Ezek után játszhatunk az előző programunkkal! Szinezzük ki kedvünkre.

Megjegyzés: a PLUS/4 Programozók Kézikönyve részletes információkat tartalmaz a többszínű grafika felhasználásáról.

8. FEJEZET

HANG ÉS ZENE PROGRAMOZÁSA

Bevezetés
Hangerő vezérlése
A zenei hang megszólaltatása
Zenei hanghatások
Zeneszerzés!
A PLUS/4 mint "zenegép"

Bevezetés

Álljon itt egy rövid program, amely a PLUS/4 zenei lehetőségeit mutatja be. Írjuk be ezt a programot, pontosan így, és futtassuk le. Megállítani pl. a RUN/STOP billentyűvel lehet.

```
10 VOL 5
20 DO
30 INPUT X
35 IF X>1015 OR X<0 THEN PRINT "0 ES 1015 KOZOTTI SZAMOT
   IRJ. .
40 SOUND 1, X, 10
50 LOOP UNTIL X=0
```

Megjegyzés: ha esetleg semmit sem hallunk, akkor állítsunk a tv-készülék hangerején.

Most nézzük meg részletesen, hogyan kell megszólaltatni egy zenei hangot a PLUS/4-en.

A hangerő vezérlése:

VOL hangerő (VOLUME)
Ez az utasítás (vagy parancs) vezérli a zenei hangok erősségét. Úgy tekinthető, mint a PLUS/4 hangerő-szabályozó gombja. A leghalkabb hangot – már nem is halljuk! – a VOL 0 adja, és a hangerő a VOL 8 hatására lesz maximális. Próbáljuk az előző példába a 10-es utasítássorban változtatni a hangerőt.

Zenei hang megszólaltatása

SOUND hang
A SOUND után mindig 3 szám vagy változó következik, amelyek mindazt közlik a számítógéppel, amit a hang lejátszásáról tudni kell.

SOUND szólam, hangmagasság, időtartam

Az első szám 1, 2 vagy 3 lehet. A PLUS/4 két különböző hangforrással dolgozik:

SOUND 1, ... az első hangforrást használja, kizárólag zenei hangokat állít elő.
SOUND 2, ... a második hangforrás által kiadott zenei hangokat adja.
SOUND 3, ... a második hangforrás egyéb hangjai, zajok, hanghatások stb. hallhatók.

Az utasításban a második szám a hangmagasságot szabja meg, tulajdonképpen a kiválasztott hang frekvenciájára utal, de nem azonos azzal! Pl. a normál "A" hanghoz tartozó 440.4 Hz-nek a 770 hangmagasság-szám felel meg. A hangmagasságot bemutató teljes táblázat a PLUS/4 ismerettárának 11. részében megtalálható, értéke 0-tól 1023-ig változhat. A szám növekedésével nő a megszólaltatott hang magassága is, 1023-nál már emberi füllel nem hallható "hangot" bocsát ki a számítógép.

Megjegyezzük, hogy ha az első szám 3-as, és a hangmagasságot beállító második számot 600 és 340 közötti tartományból vesszük, érdekes hanghatásokat érhetünk el (lásd később).

Most bemutatunk egy olyan programot, amely végigskálázza a PLUS/4 zenei hangforrását.

```
NEW
10 VOL 7
20 X=0
30 DO
40 SOUND 1, X, 50
50 X=X+5
60 LOOP UNTIL X=1023
70 VOL 0
80 END
```

Nem szóltunk még a SOUND után a harmadik számról, a hang időtartamáról. Ez a szám 0 és 65 535 közötti értéket vehet fel. A 0 beírásánál nem jön létre hang, a 65 535-nél pedig közelítőleg 16 percig szól. Tulajdonképpen egy órajelet állít be az adott számra, amelynek az értéke minden hatvanad másodpercben eggyel csökken. Így a 60-as érték 1 másodpercnek felel meg.

Zenei hanghatások

BASIC programban, ciklus utasításba építve könnyűszerrel képezhetünk különböző zenei hangokat. Pl.:

```
10 VOL 8
20 FOR S=1000 TO 500 STEP-10
30 SOUND 1, 5, 30
40 NEXT S
```

Változtassunk programunkon a magasság, időtartam és hangerőértékek módosításával.

Zajhatás

A SOUND 3,... parancs inkább zajszerű, mint zenei hangokat kelt. Az "élethű" zajok előállításához sokat kell kísérletezni. De érdemes, mert ezek a hangok különösen izgalmasak lehetnek, hiszen a kívánt zajt elemekből állítjuk össze.

A következő program egy szélvihar hangját szólaltatja meg.

```
10 VOL 2
20 R=INT (RND(0)*10) + 1
30 FOR X=1 TO R
40 SOUND 3, 600+30*X,10
50 NEXT X
60 FOR X=R TO 1 STEP-1
70 SOUND 3, 600 + 30*X,10
80 NEXT X
90 T=INT (RND(0)*100) + 30
100 SOUND 3, 600, T
110 GOTO 20
```

Tanulmányozzuk figyelmesen a program listáját. Fontos, hogy a hang magassága változó legyen, mivel a szélviharban is különböző a széllökések ereje. Ezt a 20-as és a 70-es sorokban érjük el, a véletlenszám felhasználásával. A 90-es és a 100-as programokban véletlenszerű késleltetéseket állítunk be, hogy a valóságos szélvihar egyenetlenségeinek érzetét keltsük.

Zeneszerzés

Most, miután megismertünk néhány lehetőséget, amelyekkel hanghatásokat tudunk létrehozni, egy olyan programot írunk, amely már "igazi" zenét szerez. Az 1...8 billentyűket zongorabillentyűként használhatjuk.

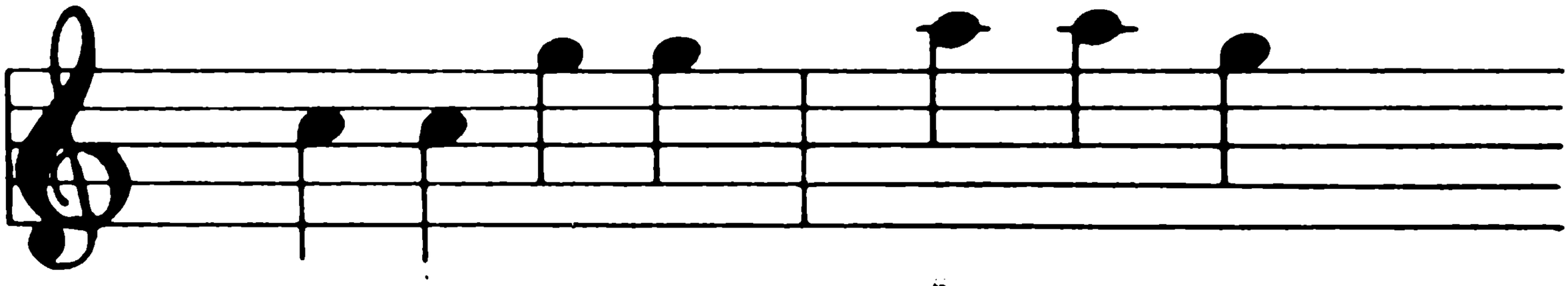
```
5 SCNCLR
10 FOR X=1 TO 8 : READ N(X): NEXT X
20 VOL 7
```

```
30 DO
40 GET A$: IFA$ = "" THEN 40
50 A = ASC(A$): IF < 49 OR A > 56 THEN 90
60 N =  A–48
70 SOUND 1, N(N), 5
80 COLOR O,N,3
90 LOOP UNTIL A = 32
100 VOL 0 : COLOR 4,2,7
110 DATA 169, 262, 345, 383, 453, 516, 571, 596
```

Bizonyára mindenki megtalálta azokat az utasításokat, amelyeket eddig még nem tárgyaltunk: READ N (X) és A = ASC(A$). Ezekre később visszatérünk, de a BASIC ismerettárban máris érdemes utánalapozni! A programnak külön érdekessége, hogy a különböző hangok elhangzásakor a képernyőkeret színe is megváltozik.

Egy egyszerű ének "kottája"

1 1 5 5 6 6 5
4 4 3 3 2 2 1
5 5 4 4 3 3 2
1 1 5 5 6 6 5
4 4 3 3 2 2 1



A következő dal programja a DATA sorok utasítása alapján hangzik el. A DATA sorban párban van a két érték, a hangmagasság és a hangzás hossza.

```
10 VOL 8
20 DO
30 READ X,Y
40 SOUND 1, X, Y
45 FOR D = 1 TO 550:NEXT
50 LOOP UNTIL X = O
60 END
100 DATA 169, 45, 169, 45, 169, 30
110 DATA 262, 15, 345, 45, 345, 30
120 DATA 262, 15, 345, 30, 383, 15
130 DATA 453, 60, 596, 45, 453, 45
```

```
140 DATA 345, 45, 169, 45, 453, 30
150 DATA 383, 15, 345, 30, 262, 15
160 DATA 169, 60
200 DATA O, O
```

A következő program különböző tempóban le és fel játszik a hangskálán, miközben különböző színek villognak a képernyőn.

```
10 VOL 8
20 DO
30 D=INT (RND(O)*5) +2:REM DURATION
40 S=INT (RND(O)*300) +700: REM START
50 R=INT (RND(O)*(1020–S)):REM RANGE
60 P=INT (RND(O)*30) +5: REM STEP
70 T=SGN (RND(1)–5):IF T=O THEN 70
80 FOR Z=S TO S+T*R STEP P*T
90 SOUND 1, Z, D
100 Y=(Z AND 15) +1:FOR X=1 TO D
110 PRINT CHR$(18);:COLOR 1,Y: PRINT " ";
120 NEXT X, Z
130 LOOP
```

A PLUS/4-es mint "zenegép"

Bemutatunk egy kicsit hosszabb programot.
Ha lenyomjuk az egyik billentyűt 1 és 9 között, egy hang elhangzik és megjelenik a képernyőn a hangjegy.

```
5 GOSUB 1000
6 FOR X=1 TO 9: READ N(X): NEXT X
8 CHAR 1, 8,1, „A NAGY ZENEGEP"
10 VOL 7
20 DO
30 GET A$:IF A$=""THEN 30
35 A=ASC(A$):IF A<49 OR A>57 THEN 50
36 N=A–48
40 SOUND 1,N(N),4
45 GSHAPE N$, 150,8*(6+(9–N),4
46 FOR Z=1 TO 50:NEXT Z
47 GSHAPE N$, 150,8*(6+(9–N),4
50 LOOP UNTIL A=32
```

```
55 VOL O:GRAPHIC O:SCNCLR
60 END
100 DATA 345, 383, 453, 516, 571, 596, 643, 685, 704
1000 GRAPHIC 1,1
1010 FOR Y=60 TO 124 STEP 16
1020 DRAW 1, 100, Y TO 200, Y
1030 NEXT Y
1040 A$="FEDCBAGFE"
1050 FOR X=1 TO 9:C=13
1060 IF INT(X/2)=X/2 THEN C=14
1070 CHAR 1, C, X+6,MID$(A$, X, 1),0
1075 CHAR 1, C+10,X+6, RIGHT$ (STR$(10–X),1)
1080 NEXT X
1090 FOR X=1 TO 8; FOR Y=11 TO 16:DRAW 1,X,Y: NEXT Y,X
1100 Y=1: X=8: DRAW 1,8,16 TO X,Y
1110 SSHAPE N$, 1, 1, 8, 16
1120 GSHAPE N$, 1, 1, 4
1130 RETURN
```

BASIC ISMERETTÁR

Ez az utolsó összefoglaló rész lexikonszerűen tartalmazza a számítógép programozásához szükséges ismereteket, amelyek kezdők, haladók, sőt szakemberek számára is hasznosak. Bizonyos részletek elolvasása alapvetően fontos a kezdőknek, pl. a BASIC 3.5 változat parancsainak és utasításainak teljes felsorolása és ismertetése. Mindenkinek ismernie kell a hibaüzeneteket, bizonyos táblázatokat. Akik már túl vannak az alapszintű programozáson, bizonyára figyelmesen tanulmányozzák majd a PLUS/4 gépi nyelvű monitorprogramjait (TEDMON), vagy az útmutatásokat a gépi kódú programozásról, a képernyő és a tár térképeket.
Ez a lexikon minden programozó számára tartogat valamit!

1. RÉSZ

Bevezetés

A kézikönyv már az előző fejezetekben is számos példát mutatott a BASIC nyelv használatára, belekóstolhattunk a programozásba, ismerkedtünk a kulcsszavak használatával. Ebben a részben egy csaknem teljes és részletes listát találunk a BASIC 3.5 nyelvről, a helyesírástól a jelentésig gyakorlóprogramokkal. Még egyszer hangsúlyozzuk, gyakoroljunk, kísérletezzünk ezekkel az új ismeretekkel bátran. A programozást megtanulni legjobban programozás közben lehet! A PLUS/4-et semmilyen programmal nem tesszük tönkre!

Az itt következő felsorolás nem arra való, hogy a BASIC nyelvet tanítsa! A parancsok és utasítások rövid leírását és példáit adja meg. A tanuláshoz a szakkönyvek listájáról válasszunk, ma már szerencsére számos ilyen témájú tankönyv is forgalomban van.

A parancsokat és az utasításokat külön részben ismertetjük, amelyen belül ezek ABC-sorrendben megtalálhatók. A parancsokat általában közvetlen üzemmódban használjuk, míg az utasításokat programban. A legtöbb parancs utasításként is használható a programban, ha sorszámmal látjuk el. Ugyanakkor jó néhány utasítás is kiadható parancsként, közvetlen végrehajtásra.

A tárgyalást a következő elvek szerint végezzük.

PARANCSOK: a számítások elvégzésére, programokkal kapcsolatos szerkesztésre, tárolásra, törlésre stb. használjuk.

UTASÍTÁSOK: számozott sorként a végrehajtásra kerülő programokban használatosak.

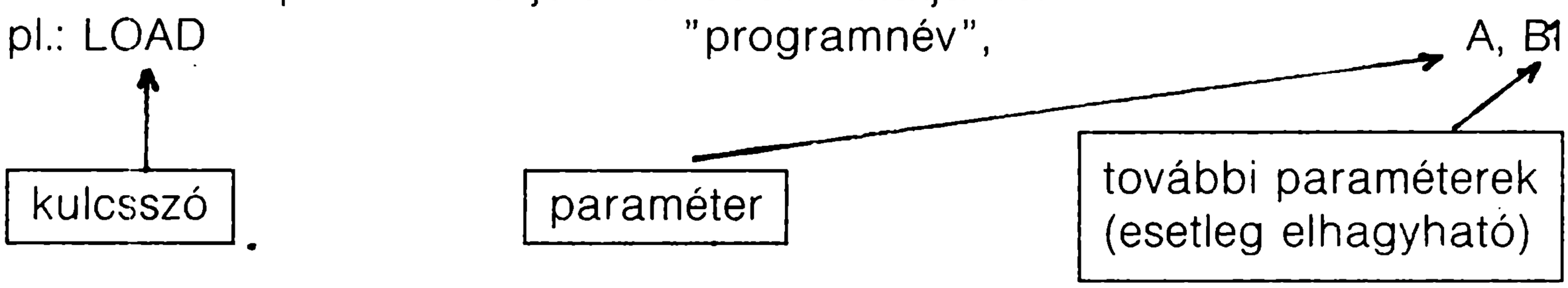
FÜGGVÉNYEK: numerikus, karakterláncot kezelő és nyomtatásra használható függvények.

VÁLTOZÓK ÉS MŰVELETI JELEK: a különböző típusú változók, a változónevek, aritmetikai és logikai műveletek.

Megjegyzés: a BASIC 3.5 parancsok és utasítások részletesebb leírása és magyarázata a PLUS/4 Programozók Kézikönyvében található. A könyv a gépi kódú programozást is tartalmazza.

A parancsok és utasítások formája

Az ismertetésre kerülő parancsok és utasítások formai megjelenítésére egységes jelölésmódot alkalmazunk, hogy a lehető legérthetőbb legyen használatuk. A legtöbb esetben rövid példákkal is segítjük a megértést.
A következő példa ezt a jelölésmódot mutatja be:



A parancsnak vagy utasításnak azt a részét, amelyet pontosan így kell beírni – pl. kulcsszó – csupa nagy betűvel írjuk. Amelyeknél ez nem lényeges – pl. programnév – kisbetűket használunk a leírásban. Ha idézőjelek is szerepelnek – pl. a program neve –, ezeket a bemutatott példának megfelelően, az adott helyre kell kiírni.
KULCSSZAVAK: a gép által ismert szavak. A parancs vagy utasítás lényegi részét jelentik, hiszen azt közlik a géppel, hogy mit kell végrehajtania. Ezeket a szavakat nem használhatjuk változónévként, és „helyesírási" hibát sem véthetünk bennük! Néhány kulcsszó helyett használható a megadott rövidítése is.
PARAMÉTEREK (ARGUMENTUMOK): ezeket kisbetűvel írva mutatjuk be. Mi választhatjuk meg, a kulcsszavakat egészítik ki a végrehajtáshoz szükséges információkkal. Egy kulcsszó közli pl. a számítógéppel, hogy töltsön be egy programot, míg a paraméter mint azonosító, azt jelöli meg, hogy melyik programot kérjük. A második paraméter ebben az esetben annak a perifériakus egységnek az azonosítója, amelyikről az adott program betölthető. Ezek a paraméterek nevek, változók, számok stb. lehetnek.
SZÖGLETES ZÁRÓJEL: a szabadon megválasztható (opcionális) azonosítókat jelölik. Ezek elhagyhatók vagy beírhatók, a parancsot, ill. utasítást mindenképpen értelmezi a számítógép.
CSÚCSOS ZÁRÓJEL: az itt felsorolt azonosítók közül egyet kell választani.
FÜGGŐLEGES VONAL: az azonosítók listáján balról elválasztja azokat, amelyeket használhatunk, és amelyeket nem. Ha szögletes zárójelen belül használjuk, akkor is csak a felsorolt listából választhatunk, de a paramétert nem kötelező beírni.
KIHAGYÁS ...: A három pont egymás után azt jelenti, hogy egy választható paraméter többször megismételhető.
IDÉZŐJELEK: karakterláncokat és más kifejezéseket határolnak. Ezek sem hagyhatók el a parancs, ill. utasítás beírásakor. Az idézőjelekkel együtt kell kiadni ezeket a karaktereket.

KEREK ZÁRÓJELEK: Beíráskor ugyanúgy használni kell ezeket a zárójeleket, mint a példákban látjuk.
VÁLTOZÓ: bármely használatos BASIC változó nevére vonatkozhat: X, A$, T% stb.
KIFEJEZÉS: bármilyen BASIC kifejezést jelenthet, pl.: A+B*2

BASIC PARANCSOK

AUTO

AUTO [szám]
Az automatikus sorszámozást indítja el, így a program beírásakor nem kell törődni az utasítássorok számozásával. Egy programsor beírása után, a RETURN nemcsak új sort jelent, hanem a következő sorszám is megjelenik. A [szám] az utasításszámok közötti növekedést jelenti. Ha az AUTO parancsot paraméter nélkül használjuk, megszűnik az automatikus sorszámozás. (Ugyanez lesz a RUN parancs hatása is.) Csak közvetlen üzemmódban használható!
Példák:

AUTO 10 : 10-esével sorszámozza a sorokat
AUTO : automatikus sorszámozás vége

BACKUP

BACKUP D meghajtó száma TO D meghajtó száma [,ON U egységszáma].

A parancsnak az a hatása, hogy a PLUS/4 átmásolja egy lemez teljes tartalmát – adatok, programok – egy másik lemezre, kettős meghajtóegység esetén. Másolhatunk új lemezre is, a HEADER parancs nélkül, mert a BACKUP parancs a lemezen levő valamennyi információt átmásolja, beleértve a formálást is. A fontosabb lemezekről ajánlatos másolatot készíteni (duplikálni) arra az esetre, ha az eredeti megsérül, elveszik stb.
A BACKUP parancs az új lemezről minden eredeti információt letöröl a másolás előtt. Ezért, ha előzetesen már használt lemezre másolunk, győződjünk meg arról, hogy nem tartalmaz-e olyan programot, amit még valaha használni akarunk. Lásd még a COPY parancsnál.
Figyelem: ez a parancs csak kettős lemez meghajtóegység esetén használható!

Példák:
BACKUP D0 TO D1 : a 0 jelű meghajtóban levő lemezről mindent átmásol az 1-ben levő lemezre
BACKUP D0 TO D1 ON U9: ugyanaz mint az előző, de a 9-es című lemez-meghajtóegységben.

COLLECT

COLLECT [D meghajtószáma] [,ON U egységszáma]

Ez a parancs arra használható, hogy felszabadítsuk a nem bevitt adatállomány által lefoglalt területet. Egyben a tartalomjegyzékből is törli az erre vonatkozó utasításokat.
Pl.: COLLECT D0

CONT

CONT (CONTINUE)

Ha a STOP billentyű, vagy a STOP, ill. END utasítással megállított program-futást akarjuk újra indítani, akkor adjuk ki ezt a parancsot. A program, onnan folytatódik, ahol a futás megszakadt, de csak akkor, ha a program soraiban semmiféle változást nem hoztunk létre. (Még a kurzort sem mozgathatjuk kész programsorra!) Természetesen programhiba esetén nem folytatódik a futás, ebben az esetben a következő hibaüzenetet kapjuk: CANT CONTINUE ERROR, azaz: hiba miatt nem folytatható.

COPY

COPY [D meghajtószáma,] "forrásállomány" TO [D meghajtószám,] "másik állomány" [,ON U egységszám]

Az egyik meghajtóegység lemezéről, a másik meghajtó lemezére másolja át a forrásállományt, feltéve, hogy kettős meghajtóegységen dolgozunk. Ha nem, akkor másolatot készít, ugyanazon a lemezen, de különböző néven. Példák:

COPY D0, "PROG" TO D1, "PROG": a PROG című programot a 0 meghajtóról átmásolja ugyanilyen néven az 1-es meghajtó lemezére.

COPY D0 TO D1: a teljes tartalmat másolja át az egyik lemezről a másikra.
COPY "JATEK" TO "JATEK": készít egy másolatot a "játék" nevű programról, ugyanarra a lemezre.

DELETE

DELETE [első sor száma] [– utolsó sor száma]

Ezzel az utasítással programsorokat törölhetünk, de csak parancs üzemmódban.
Példák:

DELETE 100 : a 100-as utasítássort törli.
DELETE 10–50 : törli az összes programsort, a 10-estől kezdve az 50-esig
DELETE – 100 : minden programsort töröl, a 100-asig bezárólag
DELETE 50 : az 50-ès programsortól kezdve az összeset törli

DIRECTORY

DIRECTORY [D meghajtószám] [, U egységszám] [, "állománynév"]

A lemez tartalomjegyzékét kiírja a képernyőre. A lista megállítható a CONTROL – S billentyűkkel, és folytatható bármelyik billentyű lenyomásával. Lassíthatjuk a kiíratást a C= billentyű lenyomásával. Ha a tartalomjegyzéket ki is akarjuk nyomtatni, akkor nem használható!
Ehhez először a tartalomjegyzéket be kell töltenünk a PLUS/4 tárolójába, de ekkor a bent levő programok törlődnek.
Példák:
DIRECTORY : kiírja a teljes tartalomjegyzéket

DIRECTORY D1, U9, "MUNKA": a "MUNKA" nevű állomány tartalomjegyzékét írja ki, amely a 9-es meghajtóegység 1-es lemezmeghajtójában van.

DIRECTORY "A" : azokat az állományokat írja ki, amelyek azonosítója A-val kezdődik

DIRECTORY D0, "?.PROGRAM": a tartalomjegyzékben azok a programok láthatók, amelyek azonosítójában, a kérdőjel helyén egy tetszőleges karakter áll. Pl.: 1. PROGRAM
Megjegyzés: ha ki akarjuk nyomtatni pl. a 8-as egységben levő 0-ás lemezmeghajtó tartalomjegyzékét, a következőket kell beírni:

```
LOAD "S0",8
OPEN 4,4: CMD4: LIST
PRINT #4: CLOSE 4
```

DLOAD

DLOAD "programnév" [, D meghajtószám] [,U egység száma]

Lemezről tölthetünk be programot ezzel a paranccsal. A program nevét mindig meg kell adni!

DLOAD "GRAFIKA" : megkeresi a lemezen a "Grafika" nevű programot, és betölti.

DLOAD (A$) : azt a programot keresi meg, és tölti be a lemezről, amelynek a neve az A$ változó értéke. Ha az üres, hibajelzést ír ki.

A DLOAD parancsot kiadhatjuk BASIC nyelvű programból is. Ezt láncolatnak nevezik, és ilyenkor a változók értéke megmarad.

DSAVE

DSAVE "programnév" [, D meghajtószám] [, U egységszám]

Ez a parancs programok tárolására szolgál, lemezen. A programot mindig el kell nevezni, mielőtt tárolnánk.
Példák:

DSAVE "KIGYO" : a „Kígyó" nevű programot lemezre másolja.

DSAVE (A$) : azt a [illegible]amot tárolja a lemezen, amelynek a neve az A$ változó értéke.

HEADER

HEADER "lemez neve" [,I ab], D meghajtószám [,ON U egységszám]

Ha egy új mágneslemezre akarunk fölvenni, vagy egy már felhasznált lemez tartalmát akarjuk törölni, akkor használjuk ezt a parancsot. A lemez neve bármilyen karaktersorozat lehet, ami nem hosszabb 16 karakternél. Az I – inicializálás – utáni két karaktert – azonosítószám – tetszőlegesen megválaszthatjuk. Figyeljünk arra, hogy minden lemeznek más azonosítót adjunk. Vigyázat! A HEADER parancs minden tárolt adatot töröl a lemezről. Ha nem adjuk meg az azonosítószámot, akkor gyors üzemmódban, a régi azonosítószámmal dolgozik a PLUS/4. Ezt a módszert csak akkor használjuk, ha úgynevezett formált lemezünk van.
Példák:

HEADER "LEMEZ", 121, D0
HEADER "DISK", IA0

HELP

A HELP parancsot akkor használjuk, ha hibaüzenetet kaptunk. Az F8 billentyűt lenyomva, a hibás sor, villogó karakterekkel megjelenik a képernyőn.

KEY

KEY [billentyűszám, karakterlánc]

A PLUS/4-esen 8 darab, a programozó által használható programbillentyű van, ebből négyet a SHIFT-tel együtt érhetünk el.
A KEY parancs kilistázza mind a nyolc billentyű funkcióját, az alapértelmezés szerint: SYS 1525, DLOAD, DIRECTORY, SCNCLR, DSAVE, RUN, LIST, HELP.
Az egyes billentyűkhöz rendelt feladatokat újra definiálhatjuk, de úgy, hogy a definíció hossza maximum 128 karakter lehet. Pl.

KEY 7, "GRAPHIC0" + CHR$(13) + "LIST" + CHR$(13)

Ennek a parancsnak a hatására parancs üzemmódban, az F7 billentyű működtetésére a gép szöveg-üzemmódba kerül (GRAPHIC 0) és kiírja a programlistát (LIST). A CHR$(13)-ra azért van szükség, mert ez a RETURN billentyű ASCII kódja. (Lásd a függelékben a kódokat.) Programból is újradefiniálhatjuk az egyes billentyűk működését, tehát a KEY utasításként is használható

```
10 KEY 2, "ELLENORZES" + CHR$ (34)
```

Ezután az F2 billentyű lenyomása az "Ellenőrzés" nevű programot fogja elindítani.
Az újradefiniált billentyűk működése visszaállítható az eredeti – alapértelmezés szerinti – állapotba, ha a RESET gombot megnyomjuk.

LIST

LIST [első sor sorszáma] [–[utolsó sor sorszáma]]

A LIST parancsra a PLUS/4 kiírja a képernyőre a tárolóban levő BASIC program utasítássorait, a sorszámoknak megfelelően. Ha a kulcsszót önmagában, paraméter nélkül használjuk, a teljes programlista megjelenik a képernyőn. A kijelzés meglehetősen gyors, ezt a C= billentyű lenyomásával lassíthatjuk, a CONTROL–S billentyűkkel pedig meg is állíthatjuk. (Újraindítás bármelyik másik billentyűvel, kivétel a RUN/STOP).

A listázás véglegesen megállítható a RUN/STOP-pal. A következő példák alapján, programunk bármelyik részét ellenőrizhetjük:
Példák:

LIST :	a teljes program listája
LIST 10 :	a 10-es utasítássor kiírása
LIST 100– :	a 100-as sortól az összes programsor
LIST –200 :	a 200-as sorig minden programsor
LIST 50–150 :	az 50-es sortól a 150-es sorig listáz

LOAD

LOAD ["programnév"] [,egység száma] [, áthelyezés]

A kazettán vagy lemezen tárolt program betöltését végzi. Ha a kulcsszót paraméter nélkül adjuk ki, a RETURN billentyű lenyomására a PLUS/4 "üzen": PRESS PLAY ON TAPE, azaz "nyomjuk meg a magnetofon lejátszógombját". Ezután feltéve, hogy a magnetofont csatlakoztattuk, a FOUND-dal jelzi a gép, ha megtalálta az első BASIC nyelvű programot. Ekkor a C= billentyű megnyomására a programot betölti a szalagról, amit a READY felirat jelez.

A LOAD parancsban, idézőjelek között, vagy sztring-változóval megadhatjuk egy bizonyos program nevét is, ekkor ezt a programot tölti be.

Az idézőjelek után, vesszővel elválasztva egy számmal, vagy numerikus változóval megjelölhetjük azt az eszközt, amelyről a betöltést kérjük. Ha ezt nem adjuk meg, automatikusan 1-nek veszi, ami a kazettás egység címe. Az áthelyezés címe, amelyet újabb vesszővel választunk el, a központi tárolóban jelöli ki a program helyét. Ennek a paraméternek 0 az értéke alapértelmezésben, és a BASIC programterület kezdetét jelenti. (A gépi kódú programok leolvasásával van igazán jelentősége.)
Példák:

LOAD : a mágnesszalagról a következő programot betölti.
LOAD "HALLO" : a mágnesszalagon megkeresi a "HALLO" nevű programot, és betölti.
LOAD A$: azt a programot olvassa le, amelynek a neve az A$ változóban van.
LOAD "HALLO",8 : a mágneslemezről betölti a "HALLO" nevű programot.

A LOAD parancsot is használhatjuk BASIC nyelvű programban, utasításként. Ilyenkor a főprogram futása közben, megkeresi, betölti és futtatja az adott programot, de a változók értéke megmarad. (Láncolat).

NEW

Ezzel törölhetjük a számítógép központi tárát: a programokat és a változók értékét is. Vigyázzunk, amikor ezt a parancsot kiadjuk!
(Programban is használható, de inkább ne használjuk!)

RENAME

RENAME [D meghajtószám,] "régi név" TO "új név" [,U egységszám]

Arra használhatjuk, hogy a mágneslemezen levő programnak új nevet adjunk. Pl.

RENAME "C16" TO "PLUS/4"

RENUMBER

RENUMBER [új kezdősor száma [, sorszámnövelés[, régi kezdősorszám]]]

A programsorok újraszámozására használjuk, csak parancs üzemmódban. Minden paraméter alapértéke 10.
Példák:

RENUMBER: Az első utasítássortól számítva, az új sorszámok: 10, 20, 30 stb.
RENUMBER 20, 20, 10: A régi sorszámozásban, az első programsor, amely a 10-es volt, most 20-es lesz, és innen kezdve 20-asával nő a sorszám
RENUMBER, 50, : Az újraszámozás után 10-ről indul és 50-esével növekszik az utasítások sorszáma.

RUN

RUN [sorszám]

A tárban levő program a RUN parancsra indul el. A végrehajtás előtt minden változó értékét törli. Ha nem írunk sorszámot, a számítógép a legkisebb számú programsorral kezdi a végrehajtást, függetlenül attól, hogy azt hová írtuk be. Ha van sorszám, akkor annál a programsornál kezdődik a futás. Programon belül utasításként is használható.
Példák:

RUN 100 : a 100-as sortól indul a program
50 RUN 100 : ha a programfutás az 50-es utasításhoz ér, a 100-as sortól indul újra a program

SAVE

SAVE ["programnév" [, egység száma [, EOT szám]]]

Ezzel a paranccsal tárolhatjuk kazettán vagy lemezen a PLUS/4-ben levő programot.

Paraméter nélkül használva, automatikusan a szalagra felveszi a programot, törli a szalagon éppen következő jeleket, és azok helyére veszi föl az új programot. Ezért legyünk figyelmesek, nehogy "rátöltsünk" egy másik

programra! Ha az idézőjelbe beírjuk a program nevét, akkor ezzel együtt veszi föl, és LOAD utasításnál hivatkozhatunk a névre. A tárolóegység száma alapértelmezés szerint 1, ez a kazettásegység címe. 8 jelenti a lemezmeghajtót. Az EOT szám (END OF TAPE) a visszatöltésnél meghatározza a program helyét. Az 1-es jelenti, hogy ugyanerre a címre töltse, ahol a SAVE előtt volt. A 2-es jelzi a "szalag végé"-t, ezután már több program nem kerülhet erre a szalagra.

Példák:

SAVE :	név nélkül a szalagra veszi a programot
SAVE "JATEK":	"Játék" néven a szalagra veszi a programot
SAVE A$:	a szalagra veszi a programot, azon a néven amely az A$ változóban található
SAVE "A",8:	lemezre másolja az "A" nevű programot
SAVE "A",1,2:	szalagon tárolja az "A" nevű programot, és "szalag vége" jelzést tesz a program végére.

SCRATCH

SCRATCH "file neve" [,D meghajtó száma] [, U egység száma]

Ez a parancs a mágneslemezről töröl egy programot. A művelet végrehajtása előtt egy kérdés jelenik meg a képernyőn: ARE YOU SURE? (Biztos ön ebben?)
Ha valóban törölni akarunk Y-t írjunk be, ha meggondoltuk a törlést N-t. Pl.:

SCRATCH "PROGRAM", D1 : ha Y-t válaszolunk, akkor az 1-es meghajtóegység lemezén törli a "Program" című programot.

VERIFY

VERIFY „programnév" [, az egység száma] [, áthelyezés jelzése]

Ez a parancs arra utasítja a PLUS/4-et, hogy a mágnesszalagon vagy a lemezen levő programot hasonlítsa össze a tárban levő programmal. Így meggyőződhetünk, hogy a programtárolás hibátlanul sikerült vagy nem, hiszen előfordulhat, hogy pl. a kazetta hibás volt az átmásoláskor. Ha a szalagon levő utolsó program nevét adjuk meg, akkor a számítógép ellenőrzi, hogy a további helyre tárolhatunk-e újabb programokat.

Abban az esetben, ha a VERIFY parancsot programnév nélkül adjuk ki, a számítógép a szalagon következő programot – tekintet nélkül a nevére – hasonlítja össze a gépben levő programmal. Ha megadjuk a parancsban a programnevet, a gép megkeresi a szalagon ezt a programot és – feltéve, hogy megtalálta – elvégzi az összehasonlítást.

A programnév után vesszővel elválasztva következhet a tárolóegység száma (kazettás adattároló 1, mágneslemez 8). Az utolsó számra ugyanaz érvényes, mint a LOAD parancsnál.
Példák:

VERIFY : a szalagon következő programot ellenőrzi
VERIFY "HALLO" : kikeresi a szalagon a "Halló" nevű programot, és összehasonlítja azzal, amelyik a tárban van.
VERIFY "HALLO", 8: megkeresi és ellenőrzi a lemezen a "Halló" nevű programot.

BASIC UTASÍTÁSOK

BOX

BOX [szín száma], a1, b1, a2, b2 [[, szög] [, festés]]

Szín: 0 és 3 közötti érték, ha nem írjuk ki, akkor alapértéke 1, ami a karakterek színét jelenti. 0 a háttér színét színét adja meg, azaz törlést eredményez
a1,b1: Az első csúcspont koordinátái (bal fölső a képernyőn).
a2, b2: az a1, b1 csúccsal szemközti csúcs koordinátái. Ha nem írjuk ki, alapértéke a grafikus pont kurzor – Pixel Cursor vagy PC – következő helyét jelenti.
szög: a középpont körüli elforgatás szögét adja meg, az óramutató járásával megegyező irányban. Ha nem adjuk meg, 0 fokot jelent.
festés: alapértéke 0, ami a "festést" kiiktatja, csak a vonalakat rajzolja. Kitölti, megfesti az ábrát, ha értéke 1.
Ez az utasítás, amely parancsként is használható, lehetővé teszi, hogy a képernyőre különböző téglalapokat rajzoljunk. Ha a paramétereknek nem adunk értéket elég ha az elválasztó vesszőket kitesszük. Ilyenkor mindenhol az alapértéket veszi a gép. Az utasítás végrehajtása után a grafikus-kurzor (Pixel Cursor) az a2, b2 pontban marad.
Példák:

a) 10 BOX 1, 10, 10 60, 60, 45, 1
Ez az utasítás egy kitöltött és 45 fokkal elforgatott négyzetet rajzol a képernyőre.
b) 10 BOX, 20, 20, 110, 140
Egy üres téglalap jelenik meg.
c) 10 BOX, 30, 90,.,45, 1 : elforgatott és kitöltött sokszöget rajzol.

CHAR

CHAR [szín], X, Y [, "karakterek"] [, inverz kijelzés]

szín: 0–3 értékű lehet. (Ugyanaz érvényes, mint a BOX utasításnál.)
X: kezdő oszlopszám, 0–39
Y: kezdő sorszám, 0–24
Karakterek: a kiíratandó szöveg, átmehet a következő sorba is.
Inverz kijelzés: alapértéke 0, azaz normál üzemmód.
Inverz kijelzés értéke, 1.

Ezzel az utasítással a képernyőn bárhol elhelyezhetjük azt a "szöveget", amelyet a karakstereknél beírtunk. A megjelenített karakterek a PLUS/4 karaktertárából (ROM) kerülnek közvetlenül a képernyőre. A kiírandó szöveget, és a kezdő pozíciót (X, Y) mindig meg kell adni, a másik két paramétert nem, miután van alapértéke.

Normál kijelzést használva – szöveg üzemmód – a CHAR utasítás úgy működik, mint a PRINT, beleértve az inverz kiírást, kurzor vezérlést stb. Ez az utasítás jól hasznosítható, ha grafikus üzemmódban, az ábrák mellett szöveget is íratunk a képernyőre.

CIRCLE

CIRCLE [szín], [a, b], X [, [Y] [, [sl] [, [sz] [, [szög] [, k]]]]

Ezzel az utasítással – parancs üzemmódban is! – háromszögeket, sokszögeket, köröket, ellipsziseket, íveket rajzolhatunk, az utóbbiakat is úgy, mint a sokszögek határesetét.
szín: 0–3, alapérték 1
a, b: a középpont koordinátái, alapértéke a grafikus-kurzor helye
X: a "kör" sugara, a képernyő vízszintes felosztása szerint
Y: a "függőleges sugár", alapértéke X-szel egyenlő
s1: a körív kezdő szöge, alapérték 0
s2: a körív befejező szöge, alapérték: 360
szög: elforgatás szöge a középpont körül. Alapértéke: 0
k: a "csúcspontok" közötti középponti szög. Alapértéke 2 fok.

Figyeljünk arra, hogy a sor és oszlop beosztás nem azonos a képernyőn. (320 × 200 grafikus pont.) Ezért a kör rajzolásakor a két sugár nem lehet azonos, mert az ellipszist adna. Az ábra annál pontosabb, minél kisebb "k" középponti szöget adunk meg!
Példák: (mindig GRAPHIC 1 vagy GRAPHIC 2!)

a) CIRCLE 1, 160, 100, 100, 80: kör, a képernyő közepén
b) CIRCLE 160, 100, 80, 40: ellipszis

c) CIRCLE, 100, 100, 40,,,,,45: nyolcszög
d) CIRCLE, 100, 50, 20, 40,,,,,90: rombusz
e) CIRCLE, 100, 100, 60,,,,,120: háromszög
f) CIRCLE, 160, 100, 50, 40,,180: félkör

CLOSE

CLOSE állomány száma

Azt az állományt, amelyet az OPEN utasításnál használtunk, lezárja. Pl. CLOSE 2.

CLR

CLR

Az összes változót törli, de a programot érintetlenül hagyja. Ezt az utasítást a RUN és a NEW parancsra a számítógép automatikusan végrehajtja.

CMD

CMD szám [, lista]

A CMD utasítás az információk kivitelét (kivitel a POKE) a képernyőről egy másik egységre küldi (pl. nyomtató). Az adott eszközt először egy OPEN utasítással meg kell nyitni, az eljárás végén CLOSE utasítással lezárni. Pl.: CMD1 a nyomtatóra irányítja a teljes kivitelt.

COLOR

COLOR kódszám, szín száma [,fényerő száma]

A kiválasztott színt az öt lehetséges kódszám alapján használja fel:
0: a háttér
1: a karakterek
2: többszínű grafika I.
3: többszínű grafika II.
4: képernyőkeret
A színeket 1–16 között választhatjuk ki, a színbillentyűknek megfelelően. (Fekete, fehér stb.). A fényerősséget 0 és 7 között állíthatjuk, bármelyik színnél, kivétel a fekete. Alapértelmezés szerint ennek 7 az értéke, ez a legnagyobb fényerő.

DATA

DATA állandók vesszővel elválasztva

A READ utasításban beolvasandó elemeket tartalmazza, amelyek számok is, karakterek is lehetnek. A karaktereket nem kell okvetlenül idézőjelek közé írni, hacsak nem tartalmaznak szóközt, kettőspontot, vesszőt vagy fölső (SHIFT) karaktert. Ha a DATA listán két vessző között semmi sincs, a READ utasítás 0-nak vagy üres karakternek olvassa be. Csak programban használható, parancs üzemmódban nem ismeri föl a számítógép! Pl.:

```
10 READ A, B, A$, B$, C, D
100 DATA 100, 25, KONYV, FUZET, 1, 2
```

DEF FN

DEF FN név (változó) = kifejezés

Ez az utasítás függvénymeghatározást jelent, lehetővé teszi, hogy bonyolult, hosszú számításokat mint matematikai függvényeket definiáljunk. Így ha egy programban többször használjuk az ilyen összetett képleteket, elég hivatkozni a függvényekre, ezzel helyet és futási időt takaríthatunk meg.
Az a név, amit a függvénynek adunk, az FN betűkkel kezdődik, a zárójelben bármilyen numerikus változó név állhat. Az = jel után írhatjuk be a kívánt képletet, amely lehet egy matematikai függvény kifejezése is. Az így meghatározott "képletre" hivatkozva, a változó helyére tetszés szerinti számot írva, a gép kiszámolja a függvény értékét. Vigyázat! Egész változókkal, ill. karakterváltozókkal nem dolgozik ez az utasítás. Pl.:

```
10 DEF FNA(X)=5*(22-A)+2*(A+1)
20 PRINT FNA (3)
RUN
103
```

DIM

DIM változó (indexek) [,változó (indexek)]...

Mielőtt a változók egy rendszerét – tömböket – használnánk, először ezzel az utasítással a tömb méretét rögzítjük, lefoglaljuk a helyet a központi tárolóban. Ha a tömbnek 11 vagy ennél kevesebb eleme van, akkor nincs szükség a helyfoglalásra.
A DIM utasítás után a tömb neve következik, amely bármilyen, helyesen megválasztott változónév lehet. A zárójelek közé írjuk az elemek számát, amely numerikus változó is lehet. Nagyszámú adatot mátrixban is elrendezhetünk, akkor a tömb többdimenziós. Az indexek száma egyben a dimenzió-

szám is. Tetszőleges dimenziót használhatunk, de figyeljünk arra, hogy az ilyen tömbök sok helyet foglalnak el a tárban. Minden tömb a 0. elemmel kezdődik, tehát pl. az A (4,4,4) háromdimenziós tömb elemeinek a száma:
$(4+1)*(4+1)*(4+1)=5\uparrow 3=125$
Egyetlen DIM utasításban több tömböt is dimenzionálhatunk, vesszővel elválasztva ezeket egymástól. Ha egy programban ugyanazt a tömböt többször is definiáljuk, hibaüzenetet kapunk. Ezért érdemes a DIM utasításokat a program elejére írni, azután nincs vele több gondunk.
Pl.:

```
10 DIM A$ (40), B (15), C (4,4,4)
```

DO/LOOP/WHILE/UNTIL/EXIT

Ennek a bonyolult utasításnak a magyar megfelelője ez lehet:
VÉGEZD (CIKLUST) MIKOZBEN (AMIG) VÉGE
Az utasítás formája:

```
DO [UNTIL kifejezés | WHILE kifejezés] utasítások [EXIT]
:
LOOP [UNTIL kifejezés | WHILE kifejezés]
```

Ennek a programrészletnek az a feladata, hogy a DO és a LOOP közötti utasításokat hajtsa végre, ciklikusan. Ha az UNTIL vagy a WHILE nem módosítja a végrehajtást, akkor ez végtelen ciklust jelent. Ha EXIT-et is használunk, akkor a végrehajtás a LOOP utasítást követő programról folytatódik. Ezek a hurkok többszörösen is egymásba ágyazhatók, hasonlóan a FOR – NEXT ciklus utasításhoz. Ha UNTIL-t használunk, a ciklus addig folytatódik, amíg az UNTIL után írt kifejezés igaz nem lesz. A WHILE éppen az ellentéte az UNTIL-nek. Az ismételt végrehajtás mindaddig fut, amíg a kifejezés igaz. A kifejezésekben az általánosan használt matematikai és logikai műveleteket használjuk =, <, >, OR (vagy), AND (és) stb.
Példák:

```
a) 10 DO UNTIL X=0 OR X=1
   :
   100 LOOP

b) 50 DO WHILE A$= "": GET A$: LOOP
```

A második példa a billentyűzetről vezérli a programfutást, mindaddig várakozik, amíg le nem nyomunk egy billentyűt.

DRAW

DRAW [szín] [,a1, b1 [TO a2,b2,,,]]

Ezzel az utasítással – parancsként is – különálló pontokat, vonalakat és idomokat rajzolhatunk. A szín száma alapértelmezés szerint 1, ekkor rajzol a grafikus kurzor, 0-nál töröl. (2 és 3 a többszínű grafikát jelenti). Az a2, b2 végpont alapértéke az a1, b1, azaz ekkor rajzol pontot. A kezdőpont alapértéke a grafikus kurzor (PC) helyét jelenti.
Példák:

a) DRAW 1, 100, 50 : pont a 100. oszlopban az 50. sorban
b) DRAW, 10,10 TO 100,100 : szakaszt rajzol
c) DRAW ,·10,10 TO 10,60 TO 60,60 TO 10,10: zárt síkidomot rajzol

END

END

Ha a programfutás END utasításhoz ér, azonnal megáll. Újraindításhoz a CONT parancsot használhatjuk.

FOR...TO..STEP

FOR ciklusváltozó = kezdőérték TO végsőérték [STEP lépésköz]

Ezt az utasítást mindig a NEXT utasítással együtt használjuk. Olyan programrészletet, amelyet többször ismételve akarunk végrehajtani, a FOR és a NEXT utasítások közé írjuk. Ha itt nincs semmi utasítás – üres ciklus – a számítógép "csak" számlálja a ciklusváltozót, amíg el nem éri a végső értéket. Így a látszat az, mintha egy rövid ideig várakozna a programfutás. A ciklus utasítás logikája a következő: először a ciklusváltozó felveszi a kezdőértéket. Amikor a programfutás a NEXT-hez ér, hozzáadja a ciklusváltozóhoz a STEP-ben megadott lépésközt – ennek is 1 az alapértéke –, és ellenőrzi, hogy az új érték nagyobb-e, mint a megadott végső érték. Ha még nem nagyobb, akkor ismét végrehajtja a FOR-t követő utasításokat, de a ciklusváltozó új értékével. Ha már túllépte a ciklusváltozó a végső értéket, akkor a NEXT után folytatódik a végrehajtás. A NEXT utasítást változó vagy változók listája követhet, vesszővel elválasztva egymástól. Ha önmagában áll a NEXT, az utoljára befejezett ciklusra vonatkozik. Pl.:

```
10 FOR I=1 TO 100
20 PRINT I
30 NEXT I
40 PRINT "KIMERULTEM! I="; I
```

Ez a program jól mutatja az utasítás végrehajtását. A képernyőn 1-től 100-ig kiírja az egész számokat, majd a szöveget, és utána a ciklusváltozó végső értékét: 101-et.

Ha a STEP paramétert megadjuk, akkor a ciklusváltozó pillanatnyi értékéhez ez hozzáadódik. Így akár visszafelé is számolhatunk – negatív lépésköz –, vagy törtszámokat is használhatunk.

Ciklusokat egymásba is építhetünk, ez az ún. egymásba skatulyázott ciklusszervezés. Ügyeljünk arra, hogy legelőször éppen az utoljára megnyitott ciklust – a legbelsőt – kell lezárni NEXT-tel! Pl.:

```
10 FOR I=1 TO 100
20 FOR J=10 TO 50 STEP 5
:
100 NEXT J
110 NEXT I
```

GET

GET változónév

A GET utasítással egyetlen karaktert olvashatunk be a billentyűzetről, a RETURN lenyomása nélkül. (Gyors adatbevitel!) A lenyomott billentyű a GET-ben megadott változó értéke lesz.

A megadott változónév lehet numerikus változó is, de ekkor csak számbillentyűt használhatunk, különben a program hibajelzéssel megáll. Ezért általában karaktervváltozót adunk meg paraméternek, így akár a RETURN billentyűt is elfogadja a PLUS/4.

Ezek utasításként, programban használhatók! Pl.:

```
100 GET A$: IF A$<>"A" THEN 100
```

A programfutás így addig várakozik, amíg az "A" billentyűt be nem nyomtuk.

GET#

GET szám, változó

Csak programon belül adhatjuk ki, és egy előzőleg már OPEN utasítással megnyitott eszközről beolvas egy karaktert. Ugyanúgy használjuk, mint a GET utasítást. Pl.:

```
10 GET#1, A$
```

GETKEY

GETKEY változónév

Nagyon hasonlít a GET utasításhoz. A programfutás addig várakozik, amíg be nem írunk egy karaktert a billentyűzetről. RETURN most sem kell! Csak programban használhatjuk a GETKEY-t is. P.:

```
10 GETKEY A$ : bármelyik billentyű lenyomására folytatódik a program
```

GOSUB

GOSUB sorszám

Ez az utasítás a megadott sorszámú utasításra adja át a vezérlést, de úgy, hogy amikor egy RETURN utasításhoz ér a programfutás, visszatér a GOSUB utáni utasítássorra. Ez így talán bonyolult, de az a lényege, hogy nem felejti el a PLUS/4, hogy a programvégrehajtásnak egy adott helyre kell visszatérnie. Az utasítás célja, a főprogramba beépített, többször ismétlődő rutinprogram hívása. Ez a szubrutin. Csak programban, és RETURN utasítással együtt használhatjuk. Pl.:

```
10 GOSUB 200
:
200 PRINT "BEMUTATKOZAS"
210 RETURN
```

GOTO

GOTO sorszám

Ez a feltétel nélküli vezérlés átadó utasítás. A program végrehajtása a GOTO-ban megadott sorszámú utasítástól folytatódik. Parancs üzemmódban is kiadhatjuk, így az adott utasítástól folytatódik a program, anélkül, hogy a változókat törölnénk. Pl.:

```
10 PRINT "PLUS/4"
20 GOTO 10
```

GRAPHIC

GRAPHIC<üzemmód [, törlés] | CLR>

Ez az utasítás a PLUS/4-et az 5 grafikus üzemmód valamelyikére állítja. Az üzemmódok:

0: normál szöveg és grafikus karakter
1: finomfelbontású grafika

2: finomfelbontású grafika, osztott képernyő
3: többszínű grafika
4: többszínű grafika, osztott képernyő

A GRAPHIC 1, ... GRAPHIC 4 utasítások egy 10 k nagyságú tárterületet jelölnek ki, és a BASIC programterület a kijelölt tárrész mögé kerül. Ez a tárfelosztás akkor is megmarad, ha GRAPHIC 0 paranccsal a szöveg üzemmódhoz (TEXT) térünk vissza. A GRAPHIC CLR parancs törli a grafikus üzemmód számára fenntartott területet.
Az utasításban, amely parancsként is használható, a képernyőtörlést az 1-es paraméter jelenti.
A GRAPHIC utasítás használatára a példákat lásd a 7. fejezetben.

IF ... THEN ... ELSE

IF kifejezés THEN utasítás [: ELSE utasítás]

Megvizsgál egy BASIC kifejezést (pl. logikai), és ha két lehetséges változata közül az egyik teljesül – igaz –, akkor a THEN utáni utasítást hajtja végre (1). Ha a kifejezés nem teljesül, azaz hamis logikai értéket vesz föl (0), a program a következő utasítássort hajtja végre, feltéve, hogy az ELSE utáni feltétel nem intézkedik máshogy. A megvizsgált BASIC kifejezés bármilyen utasítás, változó vagy művelet lehet, használhatjuk a matematikai és logikai relációjeleket (=, <, >, AND, OR, NOT).
Amennyiben használjuk az ELSE utasítást, ugyanabban a programsorban kell lennie, mint az IF...THEN-nek. Tulajdonképpen az ELSE elágazás akkor hajtódik végre, ha az IF után a kifejezés hamis. Pl.:
50 IF X>0 THEN PRINT "OK" : ELSE END

A számítógép ellenőrzi az X változó értékét. Ha nagyobb mint nulla, kiírja a képernyőre a választ, és megy a következő parancsra. Ha X nem nagyobb nullánál, a program megáll.

INPUT

INPUT ["üzenet";] változók

INPUT-tal a számítógép adatokat kér a programban szereplő változóknak, amelyeket az utasítás után egymástól vesszővel elválasztva kell felsorolni. Végrehajtáskor a programfutás addig várakozik – ezt egy ? jellel jelezve képernyőn –, míg be nem írjuk az adatokat a billentyűzetről.
Az idézőjelben szereplő üzenet kiírásra kerül, de a változóktól pontosvesszővel el kell választani.

Ha a kérdőjelre csak a RETURN gombbal válaszo'unk, a változó az eredeti értékét tartja meg.
Csak programban alkalmazhatjuk.
Példák:

```
10 INPUT "MI A NEVED" ; A$
20 INPUT "HANY EVES VAGY"; A
```

INPUT#

INPUT# szám, azonosító lista

Egy előzőleg már megnyitott eszközről olvassa le az adatokat (közvetett adatbevitel). Szöveg kijelzése tilos, csak programban fordulhat elő! Pl.:

```
10 INPUT# 1, A$ ; A
```

LET

[LET] változó = kifejezés

Ezt az utasítást nem kell használni, a PLUS/4 amúgy is végrehajtja!
Példák:

a) 10 LET X = A*B+5
b) 10 X=A*B+5

A két utasításnak pontosan ugyanaz lesz az eredménye, X egyenlő lesz A*B+5-tel.

LOCATE

LOCATE X, Y

A grafikus kurzort – PC – elhelyezhetjük a képernyőn, úgy mint egy koordináta-rendszerben. Elmozgatáskor nem látható a képernyőn. Pl.:

```
LOCATE 160, 100 :
```

a grafikus kurzor a finomfelbontású grafikus képernyő közepétől indul. (Lásd az RDOT függvény használatát.)

MONITOR

MONITOR

Ezzel az utasítással válthatunk át BASIC programokból a beépített gépi nyelvű programra.

Visszatérés BASIC–be az X billentyűvel lehetséges. (Lásd a MONITOR-program utasításait.)

NEXT

A ciklus-utasítás vége. Lásd: FOR...TO...STEP-nél.

ON

ON kifejezés < GOTO / GOSUB > első sorszám [, második sorszám ...]
Ez az utasítás úgy kezeli a GOTO és a GOSUB utasításokat, mint az IF egy speciális esetét. Az ON utasítás az adott kifejezés értékét vizsgálja meg. Ha az eredmény 1, akkor az első sorszámú utasítás végrehajtása következik, ha 2, akkor a második, és így tovább. Ha a kifejezés értéke 0, vagy nagyobb, mint amennyi sorszámot megadunk, akkor a következő utasításra ugrik a program. Negatív értéknél hibajelzést ad: ILLEGAL QU ANTITY ERROR. Pl.:

```
10 INPUT X
20 ON X GOTO 10, 40
30 PRINT " UJ SZAM" : GOTO 10
40 PRINT X
50 GOTO 10
```

OPEN

OPEN állomány száma, eszköz címe [, másodlagos cím] [,"állomány neve, típus, mód"]

Az OPEN utasításra a PLUS/4 kapcsolatot teremthet a magnetofonnal, lemezmeghajtóval, nyomtatóval.
Az állomány száma kijelöli a többi BASIC utasítás hivatkozási számát.

A 0 egységszám a billentyűzetet, 1 a magnetofont, 3 a képernyőt, 4 a nyomtatót és 8 a lemezt jelenti. A másodlagos címek különböző információkat hordoznak, pl. kazettáról olvasáskor 0, íráskor 1 stb. (Részletesen a Perifériák kézikönyvében.)

Az állomány nevét tetszőleges karaktersorozattal megadhatjuk akár szalagról, akár lemezről van szó. A típus és üzemmód viszont csak lemezre vonatkozhat (pl. írás, olvasás)
Példák:

a) 10 OPEN 3:	képernyőt megnyitja
b) 10 OPEN 15,8,15:	megnyitja a lemezen a parancs csatornát

PAINT

PAINT [szín] [,a,b] [, üzemmód]

Szín: 0–1; alapérték 1, a karakter színét jelöli ki a, b: kezdő koordináták, alapérték a grafikus-kurzor üzemmód:

0 jelenti az adott terület színezését a színparaméter által megadott színre.

1 jelenti a terület színezését a háttértől különböző színre.

A PAINT utasítással a PC (grafikus kurzor) körüli zárt területet színezhetjük ki. A művelet után a PC a kezdeti pozícióban lesz. Megjegyezzük, hogy ha a kezdőpont színe ugyanaz mint a kiválasztott szín, akkor nem láttunk színezést.

Példák:

```
10 CIRCLE, 160, 100, 65, 50
20 PAINT, 160, 100
```

A program rajzol egy kört, majd kifesti a kurzor színére.

POKE

POKE cím, érték

A POKE utasítással közvetlenül a PLUS/4 írható-olvasható tárolójába – RAM – vihetünk be adatot. Így bármit módosíthatunk a képernyőn, vagy akár változtathatunk a beviteli (kimeneti) regiszterek tartalmán is.

A cím 0-tól 65 535-ig változhat, a beírt érték pedig 0 és 255 között. Ezen értékek jelentése megtalálható a "képernyőkódok" táblázatában.

Példák:

```
10 POKE 4000, 36
```

A 4000-es címre (képernyő) tölti a 36-ot ($)

PRINT

PRINT lista

Ez a legfontosabb, leggyakrabban használt kiviteli utasítás. A lista a következőket jelentheti:

Karakterek, idézőjelek között:	"SZOVEG"
Azonosítók:	A,B%,A$
Függvények:	SIN (1), ARS (X)

Írásjelként a ; és a , használható, amelyek segítségével a képernyőn különböző módon helyezhetjük el az adatokat. A vessző négy zónára osztja fel a képernyőt, a pontosvesszővel elválasztott elemek viszont egymás mellé

kerülnek. Az írásjeleket a lista végére is kiírhatjuk, ennek az az eredménye. hogy a kiíratás úgy folytatódik, mintha az utolsó PRINT folytatódna. Parancs üzemmódban ugyanígy használható.
Példák:

a)	PRINT "COMMODORE"	COMMODORE	
	PRINT 1,2	12	
	PRINT 1;2	1	2

Ide tartoznak még a POS, SPC és TAB függvények.

PRINT #

PRINT # állomány, lista

A PRINT utasításhoz hasonlóan működik, de az állomány száma mindig egy korábban már megnyitott egységre vonatkozik. A pontosvessző és vessző használata is ugyanaz, az adatokat lemezen is elválaszthatjuk egymástól. Pl.:

```
100 PRINT # 1, A, A$
```

PRINT USING

PRINT[# állomány száma) USING "formátum"; nyomtatási lista

Ezzel az utasítással kialakíthatjuk a képernyőn, nyomtatón a megjelenés formáját. Pl.:

```
10 X=32 : Y=100.23
10 PRINT USING "$##.##"; 13.25, X, Y
RUN
$13.25$32.00$****
```

Láthatjuk, hogy amíg 13.25-öt és X értékét kiírta a 4 karakterhelyen, Y értéke helyett négy *-ot írt, mivel 5 karakternek nem adtunk formátumot. Ha Y egy karakterváltozó lett volna – pl. Y$="MACSKA" –, akkor annyi karaktert ír ki a bal oldaláról, ahány számjelet írtunk a formátumlistára.

A formátum listán különböző karaktereket használhatunk a helyfoglalásra és a nyomtatási kép definiálására. A # jel egyaránt használható számok és karakterek nyomtatásánál.

Az = és a > jeleket csak karaktereknél használhatjuk.

A plusz, mínusz, tizedespont, vessző, $, és ↑↑↑↑ csak numerikus adatok nyomtatásánál használható. A plusz és mínusz jelek a formátum első vagy utolsó pozícióján használhatóak, de mindkét oldalon nem. A plusz jel akkor jelenik meg, ha a szám pozitív. A mínusz jel pedig akkor, ha a szám negatív.

Ha mínusz jelet használunk, és a szám pozitív, a mínusz jelnek megfelelő pozíción szóköz jelenik meg. Ha numerikus adat esetén nem használunk

se mínusz, se plusz jelet a formátum mezőben, mínusz jel jelenik meg az első számjegy vagy dollár előtt, ha a szám negatív, és szóköz, ha a szám pozitív. Így tehát egy számjeggyel többet nyomtathatunk, ha a szám pozitív. Ha a # jellel és a + vagy − jelekkel kijelölt mezőbe túl sok számot akarunk írni, a mezőt csillagokkal tölti föl.

A tizedespont a formátumlistában a számban lévő tizedespont helyét jelöli. Egy formátum mezőn belül csak egy tizedespont lehet. Ha nem jelöljük ki a tizedespontot a formátum mezőben, a szám a hozzá legközelebb eső egész számra kerekítve tizedes értékek nélkül jelenik meg. Ha a tizedespontot kijelöljük, a tizedespontot megelőző jegyek száma (beleértve a mínusz jelet is, ha a szám negatív) nem haladhatja meg a # jelek számát a tizedespont előtt. Ha a szám túl hosszú, túlcsordulás keletkezik és a mező csillagokkal töltődik fel. A vessző lehetővé teszi, hogy vesszőket helyezzünk el numerikus mezőkben. A formátumlistában megadott vessző azt jelöli, hogy hol jelenjen meg a vessző a kinyomtatott számban. Csak a számon belüli vesszők jelennek meg. Az első számjegytől balra álló nem használt vesszők kitöltő karakterként szerepelnek. Legalább egy jelnek meg kell előznie a mezőben az első vesszőt. Ha vesszőket jelölünk ki a mezőben és a szám negatív, a mínusz jel jelenik meg első karakterként még akkor is, ha a karakterpozíció vesszőként van kijelölve. A megadott dollár jel hatására dollár jelenik meg a szám előtt. Ha azt akarjuk, hogy a dollár mindig közvetlenül a szám elé kerüljön, akkor legalább egy # jelet kell a dollár elé írni. Ha a dollárt vezető # jel nélkül adjuk meg, a dollár attól a pozíciótól kerül, amelyiken a formátum mezőben van.

Ha a vesszőket és/vagy plusz, ill. mínusz jelet jelölünk ki a formátum mezőben a dollárral együtt, a program vesszőt vagy előjelet ír a dollár elé.

A négy felfelé mutató nyíl segítségével egy számot tudományos jelöléssel E+ formátumban tudunk kinyomtatni. A # jelet is kell írni a négy ↑-lal együtt a mezőszélesség kijelölésére. A ↑↑↑↑-nek a # jel után kell megjelenniük a formátum mezőben. Négy ↑-at kell megadni, ha a számot E− formátumban akarjuk kiíratni (tudományos jelölés). Ha egynél több, de négynél kevesebbet írunk, szintaktikai hibát kapunk. Ha négynél többet adunk meg, csak az első négy használatára fog sor kerülni. Az ötödik és a további jelek fölöslegesek.

Egyenlőségjel használatával a karaktersorozatot a mező közepére helyezzük. A mező szélességét a # és = karakterek számával jelölhetjük ki a formátum mezőben. Ha a karaktersorozat rövidebb a kijelölt mezőnél, a mező közepén fog elhelyezkedni, míg ha több karaktert tartalmaz, mint amennyi a mezőbe fér, a jobb szélén levő karakterek csonkulnak és a karaktersorozat kitölti az egész mezőt.

A nagyobb jel a karaktermezőben való jobbra igazításra használható. A mező szélességét a # és = karakterek számával jelölhetjük ki a formátum mezőben. Ha a karakterlánc kevesebb karaktert tartalmaz, mint a mező szélessége, a karaktersorozat jobbról tölti föl a mezőt. Ha viszont több karaktert tartalmaz, mint amennyi a mezőbe fér, a jobb szélső karakterek csonkulnak, és a karakterlánc az egész mezőt kitölti.

PUDEF

PUDEF "– – – –"

Ezzel az utasítással maximum 4 szimbólumot újra definiálhatunk a PRINT USING utasításban. A szóközt, vesszőt, tizedespontot és dollár jelet valamely más karakterre cserélhetjük, úgy, hogy az új karaktert a négy hely közül a megfelelőre beírjuk a PUDEF utasításban.

Példák:

a) 10 PUDEF "*"
   a szóköz helyére * kerül a kiírásban

b) 20 PUDEF "@"
   A vessző helyett @ jelenik meg. Vigyázat, egy szóköz után írtuk be!

c) 30 PUDEF " ,"
   Egy szóköz után írtuk a pontot, azután a vesszőt. Kiíráskor a vesszők helyére pontot, a tizedespont helyére vesszőt ír a PLUS/4.

READ

READ változók listája

Ez az utasítás értéket ad a listán levő változóknak, a DATA elemei közül. A READ utasítás változói tetszőleges BASIC változók lehetnek, de összhangban kell lennie a DATA utasítás elemeivel, mert különben hibaüzenetet kapunk.

REM

REM üzenet

A programban bárhol elhelyezhető, tetszőleges üzenetet tartalmazhat – azért grafikus karaktereket inkább ne! – csak a programlistán jelenik meg. A hosszú REM üzenetek mindenesetre lelassítják a program futását.

RESTORE

RESTORE [sorszám]

Adatok újraolvasására használjuk. Sorszám nélkül az első DATA utasítás első elemétől kezdődik az újraolvasás. Sorszámmal együtt az adott sorra vonatkozik mindez.

RESUME

RESUME [sorszám I NEXT]

Arra használható, hogy folytassuk a program végrehajtását a TRAP utasítással felfedezett hiba után.
Ha nem adunk meg paramétert, akkor a hibás sortól kezdve próbálja meg újraindítani a programot. A RESUME NEXT utasításra a következő utasítástól, a RESUME N esetén pedig az N sorszámú utasítástól folytatódik a program.

RETURN

Mindig a GOSUB utasítással együtt használjuk. A RETURN után az utoljára végrehajtott GOSUB utasítást követő utasítássortól folytatódik a program. Ha nincs a RETURN előtt GOSUB, hibaüzenet jelenik meg, és megáll a program végrehajtása (RETURN WITHOUT GOSUB ERROR).

SCALE

SCALE <1/0>

A többszínű és finom felbontású grafikus üzemmódban, a szokásos bittérkép beosztását változtathatjuk meg.

SCALE 1

Többszínű üzemmódban: X=0−159 ; Y=0−199
Finom felbontású grafikus üzemmódban: X=0−319 ; Y=0−199

SCALE 0

A szokásos koordinátákat állíthatjuk vissza.

SCNCLR

Grafikus és szöveg üzemmódban is törli a képernyőt.

SOUND

SOUND hangforrás, hangmagasság, időtartam

A hangforrás száma: 1 ; zenei hang
2 ; zenei hang
3 ; hanghatások, zajok
A hangmagasság: a 0–1023 tartományból választható szám (frekvenciavezérlés).
Az időtartam: 0–65 535 között változhat, az egység a másodperc 60-ad része.

A 0 időtartam azonnal kikapcsolja a hanggenerátort. Lásd még a függelékben, a zenei hangok frekvenciaértéke táblázatát. Pl.:

SOUND 2, 800, 3600

1 percig szól, a 800-as (500 Hz) hangmagasság a 2-es hangforrásból.

SSHAPE/GSHAPE

SSHAPE karakterváltozó, a1,b1 [,a2,b2]

Ezekkel az utasításokkal a többszínű és finom felbontású képernyők derékszögű területeit tárolhatjuk, és megjeleníthetjük.
A karakterváltozóban tároljuk a terület adatait.

a1,b1: az első csúcs koordinátái
a2,b2: az a1,b1-gyel szemközti csúcs koordinátái.
(Alapérték: a grafikus-kurzor koordinátái.)
Mivel a BASIC karakterláncok hossza korlátozott, max. 255, a tárolható terület mérete is maximálva van. A tárolt alakzat a képernyőterület bármely részén megjeleníthető:

GSHAPE karakterváltozó [, [a,b] [,üzemmód]]

A karakterváltozóra és a csúcs koordinátáira ugyanaz érvényes, mint a SSHAPE utasításnál.

Az üzemmód kódja: 0–4
0: alapértelmezés, az eredeti helyre teszi az alakzatot
1: inverz megjelenítés
2: alakzat OR terület
3: alakzat AND terület
4: alakzat XOR terület

Példák:

SSHAPE " A$", 0, 0
A$ néven tárolja a képernyő 0, 0 pozíciójától a kurzor pozícióig terjedő területet.

GSHAPE " A$" ,,,1
Az eredeti alakzatot, inverz megjelenítésben kirajzolja úgy, hogy a bal fölső csúcsa a kurzorpozíciónál van.

STOP

Megállítja a végrehajtás alatt álló programot, a következő üzenettel: BREAK IN LINE N, ahol N a STOP utasítás sorszáma. CONT paranccsal fut innen tovább a program. A STOP utasítást hibakeresésnél használjuk.

SYS

SYS cím

A cím egy decimális szám 0–65 535 között. Az utasítás azt eredményezi, hogy egy gépi nyelvű programot, a megadott címtől kezdve végrehajt. Részletesen lásd a PLUS/4 programozók kézikönyvében.

TRAP

TRAP [sorszám]

Ez az utasítás kezeli a programban a hibákat, kivéve az UNDEFINED STATEMENT ERROR üzenetűeket.

A hibás sor sorszáma az EL rendszerváltozóban van, a rá vonatkozó hibakód pedig az ER rendszerváltozóban. Az ERR$ (ER) karakterváltozó tartalmazza a hibakódnak megfelelő hibaüzenetet. A vezérlés a TRAP-ban megadott sorszámú utasításra kerül.

A sorszám nélküli TRAP utasítás leállítja a hibakereső rutint.

Megjegyzés: A TRAP rutin hibáját nem lehet így megtalálni.

TRON

Ez az utasítás elindítja a nyomkövetést, minden utasítás sorszáma végrehajtáskor megjelenik a képernyőn (hibakeresés).

TROFF

Kikapcsolja a TRON utasítást.

VOL

VOL hangerő

A hangerőt 0 és 8 közötti értékre állíthatjuk. 0 jelenti a hanggenerátor kikapcsolását, 8 a maximális hangerőt. Mindkét hangforrásra vonatkozik!

WAIT

WAIT cím, 1. érték [, 2. érték]

A cím 0–65 553-ig változhat, az értékek pedig 0–255-ig. A megadott tárcím tartalma először "kizáró vagy" logikai kapcsolatba kerül a második értékkel – ha azt megadtuk a paraméterek között! –, azután pedig logikai "és" kapcsolatba az első megadott értékkel. A program csak ekkor folytatódik a WAIT utáni utasítással, ha ennek az összetett logikai műveletnek az eredménye nem nulla. Tehát a WAIT utasítással felfüggeszthető a program végrehajtása.

További ismeretek a grafika használatáról

Van még néhány olyan fogalom, amely a grafikus utasítások használatánál nagyon fontos.

Először is a PC – Pixel Cursor –, a grafikus kurzor. Mindig ott jelenik meg a következő grafikus pont, ahol a PC található. Ez mozgatható, de közben láthatatlan. A rajzoló utasításokban a valóságos X, Y koordináták helyett mindig használhatunk úgynevezett relatív koordinátákat is, amelyek a PC pillanatnyi pozíciójától számítanak. A LOCALE utasítással áthelyezhető a PC, valódi és relatív koordináták esetében egyaránt.

Példák:

a) LOCATE +100, −25: elmozgatja 100-zal jobbra és 25-tel felfelé a grafikus kurzort.

b) DRAW 1, +10, +10 TO 100, 100 : először elviszi a PC-t 10-zel jobbra és 10-zel lefelé, és azután húzza a vonalat.

c) LOCATE 50; 45: a pontosvesszővel elválasztott két szám a PC pillanatnyi helyzetéhez képest mozgatja el adott távolságra és adott szögben (50 a távolság, 45 a szög)

FÜGGVÉNYEK

Numerikus függvények

A numerikus függvényeket azért nevezzük így, mert eredményül számértéket adnak. A függvénykapcsolat a matematikai függvények kiszámításától a képernyő pozíció kijelöléséig terjed. A numerikus függvények formája a következő:

Függvény (paraméter)

A paraméter lehet szám, változó, kifejezés vagy karakter.

ABS(X) (abszolút érték)

Az abszolútérték függvény a paraméter értékét adja eredményül, ha az pozitív, és a −1-szeresét, ha negatív.

ASC(X$)
Ez a függvény az X$-ban tárolt karakterlánc első karakterének ASCII kódját adja vissza.

ATN(X) (arkusz tangens)
Megadja azt a szöget radiánban mérve, amelynek tangense X.

COS(X) (koszinusz)
A radiánban mért X szög koszinuszát adja.

DEC ("hexadecimális karakterek")
Négyjegyű hexadecimális számokat decimálisra konvertál. (0 < hexadecimális szám < FFFF) Pl.:
N=DEC ("F4")

EXP(X)
Az "e" (=2.71828183)x-edik hatványát adja.

FNxx(X)
Megadja a DEF FNxx utasításban definiált xx függvény értékét.

INSTR (1. karakterlánc, 2. karakterlánc [,kezdő pozíció])
A második karakterlánc helyzetét adja meg az első karakterláncon belül a kezdő pozíciótól kezdve. A kezdő pozíció alapértelmezés szerint az első karakterlánc kezdete. Ha nincs egyezés, 0-át ad eredményül. Pl.:

PRINT INSTR "A MACSKA A KALAPBAN VAN", "MACSKA" Az eredmény 3, mivel a "Macska" az első karakterlánc harmadik karakterén kezdődik.

INT(X) (egész rész)
Az X egész részét adja. Az eredmény mindig kisebb X-nél, vagy egyenlő vele. Így a negatív tizedes számok egész része nem azonos az eredeti szám tizedesponttól balra levő értékével. Pl.: INT (–4.5) = –5
Ha az INT függvényt fel- vagy lekerekítésre használjuk, akkor a formája a következő: INT(x + .5) vagy INT (x–.5)
Pl.:
INT(X*100 + .5)/100

JOY(n) (joystick = botkormány)
Ahol n = 1 az 1-es botkormány pozíciót, n = 2 a 2-es botkormány pozíciót jelenti.
128 vagy annál nagyobb érték a FIRE (tűz) gomb lenyomását jelzi. Az irány jelzése a következőképpen történik:

tüzelés = 128 +

		FENN		
		1		
		8	2	
BAL	7		0	3 JOBB
		6	4	
		5		
		LENN		

Pl.: Ha JOY (2) = 135-ös értéket kap, akkor a 2-es számú botkormány balra tüzel (135 = 128 + 7)

LOG(X) (logaritmus)
A függvény az X változó természetes alapú logaritmusát adja. A természetes logaritmus az e alapú logaritmus. (Lásd az EXP(X)-et). Tízes alapú logaritmusra való átszámoláshoz LOG(10)-zel kell osztani.

PEEK(X)
Ez a függvény az X tárcím tartalmát adja, ahol X, 0 és 65 535 közé esik, az eredmény pedig 0 és 255 között van, gyakran használjuk a POKE utasítással együtt.

RCLR(N)
Az N-nek adott szint adja vissza (0<N<4)
(0=háttér, 1=karakter, 2=többszínű grafika 1, 3=többszínű grafika 2, 4=keret)

RDOT(N)
A grafikus kurzor pillanatnyi pozícióját adja.

N:0 XPOS értéke (vízszintes helyzet)
N:1 YPOS értéke (függőleges helyzet)
N:2 szín

RGR(X)
Az aktuális grafikus üzemmód kódját adja meg, az X paraméter csak fiktív paraméter, értéke tetszőleges lehet.

RLUM(N)
Az N színforráshoz pillanatnyilag rendelt fényerőszintet adja meg.

RND(X)
Ez a függvény véletlen számot állít elő 0 és 1 között. Általában játékokban, statisztikai feladatokban, véletlenszerű események szimulálására használjuk.
Az első véletlen számot érdemes RND (–TI) formában előállítani, így a véletlen számok mindig más alapszámból indulnak. Ezután X-nek érdemes az 1 értéket adni, így a véletlen számok előállítása mindig a korábbi X értéken alapul, amelyet egy belső függvény számol.

Pl.: kockadobást szimulál a következő utasítás:

10 A= INT (RND(1)*6+1)
Azért szorzunk 6-tal, hogy a 0 és 1 közötti véletlen szám 0 és 6 közé kerüljön. Így 1-et hozzáadva, és a törtrészt levágva az INT függvénnyel éppen 1 és 6 közötti egész számokat kapunk.
Példák:

a) 100X= INT(RND(1)*1000)+1
   1 és 1000 közötti egész számokat ad, véletlenszerűen.
b) 200X= INT(RND(1)*150)+100
   100 és 249 között ad véletlen számokat.

SGN(X)
Ez a függvény az X előjelét adja meg. Ha X pozitív, az értéke 1, ha X negatív –1, és X=0, akkor SGN(0)=0

SIN(X) (szinusz)
Ez a függvény a trigonometrikus szinusz függvény. Az X szöget radiánban értelmezi.

SQR(X)
Megadja X négyzetgyökét, ha X nem negatív. Negatív X-nél hibaüzenetet kapunk:

ILLEGAL QANTITY ERROR

TAN(X) (tangens)
Az X szög tangensét adja meg, a szöget radiánban értelmezi.

USR(X)
Ezt a függvényt használva, a programvégrehajtás a gépi kódú programra ugrik. (Kezdőcíme a 1281 és 1282 tárcímeken található.) Így lehetővé válik az adatcsere a BASIC és a gépi kódú programok között.
A gépi kódú programokról részletesen a COMMODORE PLUS/4 Programozók Kézikönyvében van szó.

VAL(X$)
Ez a függvény az X$-ban levő karaktersorozatot vizsgálja, és balról jobbra haladva az első érvénytelen numerikus karakterig számmá alakítja.
Példák:

a) X= VAL ("123.456")	X= 123.456
b) X= VAL ("39 A 17")	X= 39
c) X= VAL ("17 E 02")	X=1700
d) X= VAL ("A 293")	X=0

Sztringkezelő függvények (karakterfüggvények)

Ezek a függvények karakterekből számokból vagy idézőjelek közé írt karakterláncból állítanak elő számjegyeket, karaktereket, vagy grafikus jeleket.

CHR$ (X)
Az a karakter lesz a függvény értéke, amelynek az ASCII kódja X

ERR$ (N)
Lásd a TRAP utasításnál

HEX$ (N)
Egy négyjegyű karaktersorozatot állít elő, amely az N decimális szám hexadecimális megfelelője. N csak 0 és 65 535 közé eshet.

**LEFT$** (X$, X)
Az X$ karakterváltozó bal oldali X karakterét adja eredményül. Pl.:

```
10 X$ = "PLUS/4"
20 A$ = LEFT$ (X$, 2)
30 PRINT A$
RUN
PL
```

LEN (X$)
Megadja az X$ karakterlánc karaktereinek a számát, ebbe a szóközöket is beleszámolja.

**MID$** (X$, S, X)
A függvény értéke egy X karakterből álló karaktersorozat lesz, amely az X$ karakterlánc S-edik karakterével kezdődik.
Állhat egy értékadó utasítás bal oldalán, és használhatjuk függvényként is. Jól használható karakterek kiválasztására, és karakterek cserélésére is.
Pl.: MID$ ("ABCDE",3,2) értéke: CD

**RIGHT$** (X$, X)
Az X$ jobb oldali X karakteréből álló karaktersorozat lesz az eredménye.

STR$ (X)
X-et mint karakterláncot állítja elő, de úgy, ahogy az megjelenítésre kerülne.

Egyéb függvények

FRE (X)
Megadja a szabad tárterületet bájtokban. (1 bájt = 8 bit) X csak látszólag paraméter.

POS (X)
Annak az oszlopnak a számát adja meg a képernyőn – 0 és 79 között –, ahol a következő PRINT utasítás kezdődhet.

SPC (X)
A PRINT utasításon belül X szóköz átugratását adja meg. X csak 0 és 255 közötti egész szám lehet.

TAB (X)
PRINT-ben használjuk, azt jelenti, hogy a következő kiírás az X számú oszlopban kezdődik. Az X értéke 0 és 255 közötti szám.

π (pi)
Kifejezésben használva 3.14159265 lesz az értéke.

Változók és műveleti jelek

A PLUS/4 három típusú BASIC változót használ: numerikus valós, numerikus egész és karakterváltozó.

Numerikus valós változók

Szokás normál vagy lebegőpontos változóknak is nevezni. Az abszolút értékük 10^{-38} és 10^{38} közötti szám lehet, 9 számjegy pontossággal. Az ennél pontosabb, vagy több számjegyből álló számok (pl. 1234567890) kijelzésénél a gép normál alakra tér át. Tehát pl. az A = 1234567891 számot A = 1.23456789E + 09 alakban írja ki, tehát csak 8 tizedesjegy pontosságig, mivel egy helyet az egész résznek, egyet pedig a tizedespontnak tart fent.

Numerikus egész változók

A –32768 és a +32767 számtartományban levő egész számokat vehetik föl értékül.
Az egész típusú változók és számok kisebb tárhelyet foglalnak el, mint a valósak.

Karakterváltozók

Bármely, a PLUS/4 által használt karaktert tárolhatnak.

Változónevek – azonosítók

A változók azonosítóiból csak az első két karaktert használja a számítógép. Ez lehet egyetlen betű, egy betű és egy szám, vagy két betű.
Az egész változókat % jellel azonosíthatjuk, a karakterváltozókat pedig a $-jellel.
Példák:
a) A,B5,CD : numerikus valós változók azonosítói

b) A%,B1%: numerikus egész változók azonosítói
c) A$,PL$: karakterazonosítók

Tömbök – tömbváltozók

A tömbök tulajdonképpen ugyanazon név alatt szereplő változók sorozatát tartalmazzák, amelyen belül az egyes változókat – a tömb elemeit – külön szám azonosítja. Mind a három típusú változók tömbbe rendezhetők, ha a program úgy kívánja, de előzőleg a tömbnek helyet kell biztosítani a tárban a DIM utasítással. Pl.:

```
DIM A(7), B%(10), A$(40)
```

A tömbök nemcsak egy dimenziósak is lehetnek. Pl. a kétdimenziós tömb elemei egy koordináta-rendszer pontjainak a koordinátái lehetnek. Ilyen esetben két számot kell megadni, az első a sorokat, a második az oszlopokat adja meg. Pl.:

```
A(7,2), B%(3,3,2)
```

Rendszer-azonosítók (fenntartott változónevek)

Hét olyan azonosító van, amelyeket a PLUS/4 saját céljaira tart fenn: DS, DS$, ER, EL, ST, TI, TI$.
Ezeket nem használhatjuk a programokban változónévként, hasonlóan a BASIC kulcsszavakhoz. Sőt olyan azonosító sem szerepelhet, amely tartalmazza valamelyik kulcsszót. Pl. ARUN, BLOAD stb.

ST: státusz változó. Az értéke az előzőleg végrehajtott beviteli/kiviteli művelet eredményétől függ. (Kivétel a normál képernyő és billentyűzet műveletek.) Ha az adatmozgatási művelet rendben megtörtént, ST értéke 0.

TI, TI$: A PLUS/4 jelfrissítő órajelére vonatkoznak. Ez egy valódi órajel, amely bekapcsoláskor 0-ról indul, és minden 1/60-as másodpercben lép egyet. A TI$ változó értékének megváltoztatásával nullázható az órajel, és ez a változó 24 órás beosztásban tartalmazza a pillanatnyi időértéket. A TI$ első két karaktere az órákat, a harmadik és negyedik a perceket, az utolsó kettő a másodperceket jelenti. Ha eltelt 23 óra, 59 perc és 59 másodperc – vagy a PLUS/4-et kikapcsoljuk – a változó értéke nullázódik. Pl.:

```
TI$ = "081530" beállítja az órát 8 óra, 15 perc, 30 másodpercre.
```

DS: a lemezmeghajtó egység pillanatnyi helyzetét adja meg. Értékét a PRINT DS$ utasítással megjeleníthetjük. Ha DLOAD vagy DSAVE parancs után villog a piros fény, akkor érdemes használni.

ER, EL: csak programon belül használhatók, hibaelemző rutinokban. Az ER változó az utolsó hiba számát tartalmazza, ameddig a program futott. Az EL azt a sorszámot, ahol az utolsó hiba előfordult.

A PRINT ERR$/ER/ kiírja az adott hibaüzenetet.

Basic műveletek

Aritmetikai műveletek:

\+ : összeadás
– : kivonás
* : szorzás
/ : osztás
↑ : hatványozás
Ha egy utasításban zárójel használata nélkül, több műveleti jel is előfordul, a végrehajtás a prioritási sorrendben történik: hatványozás, szorzás és osztás, összeadás és kivonás. Az azonos "fontosságú" műveletek végrehajtása ilyenkor balról jobbra történik. Ezt a sorrendet zárójelek használatával megváltoztathatjuk, ilyenkor a gép először a zárójelben kijelölt műveleteket végzi el. Ügyeljünk, hogy balról ugyanannyi zárójelet használjunk, mint jobbról. (Különben: SYNTAX ERROR üzenetet kapunk.)

Relációs műveleti jelek

= : egyenlő
< : kisebb mint
> : nagyobb mint
<= : kisebb vagy egyenlő
>= : nagyobb vagy egyenlő
<> : nem egyenlő
Az aritmetikai műveletek mindig magasabb prioritásúak, mint a relációs műveletek.
Három logikai művelet is használható, ezeknek a prioritása a legalacsonyabb:

AND : logikai ÉS
OR : logikai VAGY
NOT : logikai NEM

Ezeket leggyakrabban aritmetikai kifejezések egyesítésére használjuk, pl. IF...THEN utasításban. Pl.:

```
IF A=B OR C>D THEN 100
```

2. RÉSZ

BASIC 3.5 rövidítések

Lehetőségünk van az utasítások, ill. parancsok beírásának lerövidítésére. Ennek a módszerét ismerhetik meg a következőkben. A kulcsszavaknak megfelelő rövidítések úgy értendők, hogy a nagy betűt mindig SHIFT-tel írjuk be.
Pl.: "aS" azt jelenti, hogy kisbetűként az "a" billentyű, utána SHIFT-tel együtt az S billentyűt.
A "típus" a kulcsszavakról ad rövid információt. Azok a kulcsszavak, amelyek nem találhatók meg a táblázatban nincs rövidítésük.

Kulcsszó	Rövidítés	Típus
ABS	aB	numerikus függvény
AND	aN	logikai művelet
ASC	aS	numerikus függvény
ATN	aT	numerikus függvény
AUTO	aU	parancs
BACKUP	bA	parancs
BOX	bO	utasítás
CHAR	chA	utasítás
CHR$	cH	karakterfüggvény
CIRCLE	cI	utasítás
CLOSE	clO	utasítás
CLR	cL	utasítás
CMD	cM	utasítás
COLLECT	colL	parancs
COLOR	colO	utasítás
CONT	cO	parancs
COPY	coP	parancs
DATA	dA	utasítás
DEF FN	dEfn	utasítás
DELETE	deL	parancs
DIM	dI	utasítás
DIRECTORY	diR	parancs
DLOAD	dL	parancs
DRAW	dR	utasítás

Kulcsszó	Rövidítés	Típus
DSAVE	dS	parancs
END	eN	utasítás
ERR$	eR	karakterfüggvény
EXIT	exI	utasítás
EXP	eX	numerikus függvény
FOR	fO	utasítás
FRE	fR	numerikus függvény
GET	gE	utasítás
GETKEY	gEKe	utasítás
GOSUB	goS	utasítás
GOTO	gO	utasítás
GRAPHIC	gR	utasítás
GSHAPE	gS	utasítás
HEADER	heA	parancs
HELP	heL	parancs
HEX$	hE	karakterfüggvény
IF...GOTO	if...gO	utasítás
IF...THEN...ELSE	if...tH...eL	utasítás
INPUT#	iN	utasítás
INSTR	inS	numerikus függvény
JOY	jO	numerikus függvény
KEY	ke	parancs
LEFT$	leF	karakterfüggvény
LIST	lI	parancs
LOAD	lO	parancs
LOCATE	loC	utasítás
LOOP	loO	utasítás
MID$	mI	karakterfüggvény
MONITOR	mO	utasítás
NEXT	nE	utasítás
NOT	nO	logikai művelet
ON...GOSUB	on...goS	utasítás
ON...GOTO	on...gO	utasítás
OPEN	oP	utasítás
PAINT	pA	utasítás
PEEK	pE	numerikus függvény
POOK	pO	utasítás
PRINT	?	utasítás
PRINT#	pR	utasítás
PRINT USING	? usI	utasítás
PRINT# USING	PR uSI	utasítás

Kulcsszó	Rövidítés	Típus
PUDEF	pU	utasítás
RCLR	rC	numerikus függvény
RDOT	rD	numerikus függvény
READ	rE	utasítás
RENAME	reN	parancs
RENUMBER	renU	parancs
RESTORE	reS	utasítás
RESUME	resU	utasítás
RETURN	reT	utasítás
RGR	rG	numerikus függvény
RIGHT$	rI	karakterfüggvény
RLUM	rL	numerikus függvény
RND	rN	numerikus függvény
RUN	rU	parancs
SAVE	sA	parancs
SCALE	scA	utasítás
SCNCLR	scN	utasítás
SCRATCH	scR	parancs
SGN	sG	numerikus függvény
SIN	sI	numerikus függvény
SOUND	sO	utasítás
SPC(	sP	függvény
SQR	sQ	numerikus függvény
SSHAPE	sS	utasítás
STEP	stE	utasítás
STOP	sT	utasítás
STR$	stR	karakterfüggvény
SYS	sY	utasítás
TAB(	tA	függvény
TRAP	tR	utasítás
TROFF	troF	utasítás
TRON	trO	utasítás
UNTIL	uN	utasítás
USR	uS	speciális függvény
VAL	vA	numerikus függvény
VERIFY	vE	parancs
VOL	vO	utasítás
WAIT	wA	utasítás
WHILE	wH	utasítás

3. RÉSZ

Fordítóprogramok

Standard BASIC programok átalakítása COMMODORE BASIC 3.5-re

Amennyiben más típusú számítógépen megírt BASIC programot szeretnénk a PLUS/4-esen futtatni, néhány kisebb átalakításra van szükség. Ebben a részben néhány jó tanácsot közlünk az átírás megkönnyítésére.

Átalakító Programok

Minden olyan utasítást töröljünk, amely a karaktertváltozók hosszúságát jelöli ki.
A DIM A$ (I,J) utasításokat, amely egy-egy tömböt I hosszúságú J elemre dimenzionál át kell írni így:

```
DIMA$ (J)
```

Néhány BASIC változat vesszőt és egyéb jeleket használ a karakterláncok összefűzésére. Ezeket + jelre kell változtatnunk, a COMMODORE BASIC-ben ez jelenti az összefűzést.
A BASIC 3.5-ben a karakterláncok részeit csak a MID$, LEFT$, RIGT$ függvényekkel vihetjük ki. Az X$ = A$ (I) vagy X$ = A$ (I,J) utasításokat a következőképpen kell megváltoztatni:
X$ = A$(I) helyett: X$ = MID$ (A$,I,1)
X$ = A$(I,J) helyett: X$ = MID$ (A$,I,J)

Többszörös hozzárendelések

Néhány számítógépen a B-nek és a C-nek 0 értéket adhatunk a következő utasítással

```
10 LET B=C=0
```

A C = BASIC-ben ezt nem használhatjuk, helyette:

```
10 B=0 : C=0
```

Többszörös utasítások

Egy programon belül, az egyes utasítások elkülönítésére csak kettőspontot használhatunk.

Matematikai függvények

A matematikai függvényeket csak a BASIC 3.5 ismerettárban megadott formában használhatjuk, ábrázolásukhoz FOR...NEXT ciklust használjuk.

Az F billentyűk újraprogramozása

A programbillentyűk eredeti működését megváltoztathatjuk, hogy más COMMODORE gépekkel megegyezzenek. Így pl. programok átmásolása is egyszerűbbé tehető.
Az újraprogramozáshoz írjuk be a PLUS/4-be a következő programot:

```
10 FOR A=1 TO 8 : READ B: KEY A, CHR$(132+B) :
NEXT : DATA 1,5,2,6,3,7,4,8
RUN
```

Ezek után az F billentyűk lenyomására egy 133 és 140 közötti karakter jelenik meg, hasonlóan a COMMODORE 64 működéséhez.
A következő programmal kiegészítve ellenőrizhetjük mindezt:

```
20 GETKEY A$ : IF ASC (A$)=133 THEN PRINT "F1 BILLENTYÜ":
   GOTO 20
30 IF ASC (A$)>133 AND ASC (A$)<141 THEN PRINT
   "EGY MASIK F BILLENTYÜT LENYOMNI"
40 GOTO 20
```

Ha már nincs szükségünk az újraprogramozásra, ki kell kapcsolni a PLUS/4-est. Visszakapcsolás után az F billentyűk eredeti működésük szerint dolgoznak.

4. RÉSZ

Hibaüzenetek

Ezeket a hibaüzeneteket a BASIC írja ki. A hibaüzenetek az ERR$() függvény alkalmazásával is kiírathatók. A hibakód a hibára vonatkozó szám, amit a függvény alkalmazásakor megadunk.

Hibakód	Hibaüzenet	
1	**TOO MANY FILES** (túl sok állomány van nyitva)	Egyidejűleg maximum 10 állomány lehet megnyitva.
2	**FILE OPEN** (az állomány meg van nyitva)	Kísérlet egy már megnyitott állomány újbóli megnyitására.
3	**FILE NOT OPEN** (az állomány még nincs nyitva)	A kimeneti/bemeneti utasításban szereplő állományt használat előtt meg kell nyitni.
4	**FILE NOT FOUND** (ilyen állomány nincs)	Az adott nevű állomány nem létezik (lemezen).
5	**DEVICE NOT PRESENT** (ilyen egység nincs)	A megadott kimeneti/bemeneti egység nem érhető el.
6	**NOT INPUT FILE** (nem bemeneti állomány)	GET-tel vagy INPUT-tal megkísérelt adatbevitel kimeneti állományból.
7	**NOT OUTPUT FILE** (nem kimeneti állomány)	Megkísérelt adatkivitel csak bemenetiként kezelt állományba.
8	**MISSING FILE NAME** (hiányzó állománynév)	Lemezre kiadott OPEN, LOAD vagy SAVE utasításokban általában szükség van az állománynévre.
9	**ILLEGAL DEVICE NUMBER** (érvénytelen egységcím)	Helytelen eszközhasználatra tett kísérlet. (SAVE a képernyőre stb.)
10	**NEXT WITHOUT FOR** (NEXT nélküli FOR)	Ciklusok hibás egymásba ágyazása, vagy a NEXT utasításban megadott változó nem egyezik a FOR-ban megadottal.
11	**SYNTAX** (szintaktikai hiba)	A BASIC nem ismeri fel az utasítást. Hiányzó vagy fölösleges zárójelek, elírt kulcsszavak stb. miatt.
12	**RETURN WITHOUT GOSUB** (RETURN GOSUB nélkül)	Egy RETURN utasítás végrehajtására került sor előzetes GOSUB nélkül.

13 **OUT OF DATA** (túl kevés adat)	A READ utasítás nem talál több beolvasásra szánt adatot.
14 **ILLEGAL QUANTITY** (érvénytelen mennyiség)	Egy függvény vagy utasítás argumentumaként megadott szám kívül esik a megengedhető tartományon.
15 **OVERFLOW** (túlcsordulás)	Egy számítás eredménye nagyobb a legnagyobb ábrázolható számnál (1.701411833E+38).
16 **OUT OF MEMORY** (nincs szabad memória)	Nincs szabad tárterület a program vagy a változók tárolására, illetve túl sok DO, FOR vagy GOSUB utasítás van érvényben.
17 **UNDEF'D STATEMENT** (nemlétező utasítás)	Egy nem létező utasítássorra való hivatkozás.
18 **BAD SUBSCRIPT** (helytelen tömbindex)	Olyan tömbelemre való hivatkozás, mely a DIM utasításban megadott határon kívül esik.
19 **REDIM'D ARRAY** (tömb újra dimenzionálása)	Egy tömb csak egyszer dimenzionálható. Ha egy tömbre dimenzionálása előtt hivatkozunk, mérete automatikusan DIM (10)
20 **DIVISION BY ZERO** (nullával való osztás)	A nullával való osztás érvénytelen.
21 **ILLEGAL DIRECT** (érvénytelen parancs)	Az INPUT vagy GET utasítások csak programban használhatók.
22 **TYPE MISMATCH** (nem egyező típus)	Szám szerepel karakter helyett vagy fordítva.
23 **STRING TOO LONG** (túl hosszú karakterlánc)	Egy karakterlánc maximum 255 karaktert tartalmazhat.
24 **FILE DATA** (hibás adat)	Hibás adat olvasása szalagról.
25 **FORMULA TOO COMPLEX** (túl bonyolult képlet)	Egyszerűsíteni kell a kifejezést (két részre bontani vagy kevesebb zárójelet alkalmazni).
26 **CAN'T CONTINUE** (nem folytatható)	A CONT parancs érvénytelen, ha a program még nem futott, hiba fordult elő vagy egy sort módosítottak.
27 **UNDEF'D FUNCTION** (nem definiált függvény)	Felhasználó által definiált függvényre való hivatkozás a függvény definiálása előtt.
28 **VERIFY** (eltérés)	A lemezen vagy szalagon lévő program nem egyezik a memóriában lévővel.

29	**LOAD** (betöltési hiba)	Betöltés közben föllépett hiba.
30	**BREAK** (megszakítás)	A program végrehajtása a stop billentyűvel megszakítva
31	**CAN'T RESUME** (TRAP nélküli RESUME)	TRAP utasítás kiadása nélkül előfordult RESUME
32	**LOOP NOT FOUND** (LOOP nélküli DO)	A programban előfordult DO utasításhoz nem tartozik LOOP
33	**LOOP WITHOUT DO** (DO nélküli LOOP)	Aktív DÒ utasítás nélkül előfordult LOOP
34	**DIRECT MODE ONLY** (csak parancsmódban)	A parancs csak közvetlen módban használható, programban nem.
35	**NO GRAPHICS AREA** (nincs grafikus terület lefoglalva)	Grafikus utasítások (DRAW, BOX stb.) előfordulása előzetes GRAPHIC utasítás nélkül.
36	**BAD DISK** (rossz lemez)	Lemezinicializálási hiba a gyors (ID nélküli) módszer előzőleg nem formált lemezen való alkalmazása vagy hibás adathordozó miatt.

Lemez-hibaüzenetek

Ezeket a hibaüzeneteket a DS és DS$ fönntartott változók segítségével lehet megjeleníteni.
Megjegyzés: a 20-nál kisebb hibaüzenetek elhanyagolhatók a 01 kivételével, mely a SCRATCH utasítással törölt állományok számára vonatkozik.

20	**READ ERROR**	A lemezvezérlő nem tudta azonosítani a keresett adatblokk fejlécét. Az ok érvénytelen szektorszám vagy a fejléc sérülése lehet.
21	**READ ERROR**	A lemezvezérlő nem talált szinkronjelet a keresett sávon. Oka lehet az író/olvasó fej helytelen beállítása, az adathordozó hiánya, egy nem formált vagy helytelenül behelyezett lemez. Jelezhet géphibát is.
22	**READ ERROR**	A lemezvezérlő egy nem megfelelően felírt adatblokkot olvasott vagy ellenőrzött. Ez a hibaüzenet a BLOCK parancsok használatakor fordul elő, és érvénytelen sáv/szektor olvasását jelzi.

23 **READ ERROR**	Ez az üzenet azt jelzi, hogy egy vagy több adatbyte hibás. Az.adatok beolvasása a DOS tárjába megtörtént, de az adatok után beolvasott ellenőrző összeg hibás. Utalhat földelési hibára is.
24 **READ ERROR**	Az adat vagy fejléc beolvasása a DOS tárjába megtörtént, de egy hardver hiba az adat-mezőben érvénytelen bitkombináció beolvasását jelzi.
25 **WRITE ERROR**	Ez az üzenet akkor fordul elő, ha a vezérlő eltérést talál a felírt és a DOS tárjában levő adatok között.
26 **WRITE PROTECT ON**	Ez az üzenet akkor fordul elő, ha a vezérlő írás utasítást kapott, miközben a lemezen az írás-engedélyező rész le van ragasztva.
27 **READ ERROR**	A vezérlő a keresett adatblokk fejlécében hibát talált. Az adat beolvasására nem kerül sor. Utalhat földelési hibára is.
28 **WRITE ERROR**	A vezérlő egy adatblokk felírása után a következő fejléc szinkronjelére vár. Ha a szinkronjel nem jelenik meg az előre megadott időn belül, ez a hibaüzenet fordul elő. A hiba rossz lemezformálás (az adat belenyúlik a következő blokkba) vagy hardver hiba miatt lép föl.
29 **DISK ID MISMATCH**	Ez az üzenet akkor fordul elő, ha a vezérlő egy nem inicializált lemezt akar elérni. A hiba akkor is fölléphet, ha a lemez fejléce hibás.
30 **SYNTAX ERROR**	A DOS a parancscsatornán küldött parancsot nem tudja értelmezni. Rendszerint érvénytelen számú vagy formájú állománynév megadásakor fordul elő. Például két állománynév szerepel a COPY parancs bal oldalán.
31 **SYNTAX ERROR**	A DOS nem ismeri föl a parancsot. A parancsnak az első oszlopban kell kezdődnie.

32 **SYNTAX ERROR**	A kiküldött parancs 58 karakternél hosszabb.
33 **SYNTAX ERROR**	Az OPEN vagy SAVE parancsban érvénytelen paraméterek szerepelnek.
34 **SYNTAX ERROR**	A parancsból hiányzik az állománynév, vagy a DOS nem ismeri föl. Rendszerint a kettőspont (:) hiányzik a parancsból.
39 **SYNTAX ERROR**	Akkor fordulhat elő, ha a parancscsatornán (a másodlagos cím 15) küldött parancsot a DOS nem ismeri föl.
50 **RECORD NOT PRESENT**	Az utolsó rekordon túl INPUT# vagy GET# parancsokkal megkísérelt olvasás eredménye. Az üzenet a relatív állomány vége utáni pozicionáláskor is előfordul. Ha a cél az állomány kiterjesztése új rekordok hozzáadásával (a PRINT# paranccsal), a hibaüzenet figyelmen kívül hagyható. A hiba fellépésekor INPUT vagy GET parancsot csak újrapozicionálás után lehet kiadni.
51. **OVERFLOW IN RECORD**	A PRINT# parancs végrehajtáskor túllépi a rekord határát. A ki nem írt adat elvész. Mivel a rekord végjeleként küldött kocsi vissza is beleszámít a rekord méretébe, a hiba akkor fordul elő, ha a rekord összes karaktereinek száma (beleértve a záró kocsi visszát), túllépi a megadott méretet.
52 **FILE TOO LARGE**	Relatív állományon belül a rekord pozíciója a lemez túlcsordulását eredményezi.
60 **WRITE FILE OPEN**	Az üzenet egy írás után le nem zárt állomány olvasás előtti megnyitásakor fordul elő.
61 **FILE NOT OPEN**	Az üzenet egy meg nem nyitott állomány megkísérelt elérésekor fordul elő. Az üzenet nem mindig jelenik meg, csak a végrehajtásra egyszerűen nem kerül sor.
62 **FILE NOT FOUND**	A keresett állomány nem található a jelzett meghajtón.
63 **FILE EXISTS**	A létrehozni kívánt állomány már szerepel a lemezen.

64 **FILE TYPE MISMATCH**	A keresett állomány típusa nem egyezik a tartalomjegyzékben szereplő állomány típusával.
65 **NO BLOCK**	Ez az üzenet a B-A parancs használatakor fordul elő. Azt jelzi, hogy a lefoglalandó blokk már foglalt. A paraméterek a következő szabad sáv és szektor címét tartalmazzák. Ha a paraméterek értéke 0, minden magasabb számú blokk foglalt.
66 **ILLEGAL TRACK AND SECTOR**	A DOS a formátumban nem létező sáv vagy blokk elérését kísérelte meg. A következő blokk mutatójának olvasási hibája eredményezheti.
67 **ILLEGAL SYSTEM T OR S**	Ez a különleges hibaüzenet a rendszer által használt érvénytelen sáv- vagy szektor-címet jelez.
70 **NO CHANNEL**	A keresett csatorna nem érhető el, vagy minden csatorna használatban van. Egyidejűleg maximum 5 soros állomány lehet megnyitva a DOS-ban. A közvetlen elérésű csatornák hat nyitott állomány elérését teszik lehetővé.
71 **DIRECTORY ERROR**	A BAM nem egyezik a belső számlálóval. A hiba a BAM foglaltságában van, vagy törlődött a DOS memóriájában. A hiba kiküszöböléséhez helyre kell állítani a BAM-ot a memóriában, az I parancs végrehajtásával. A hiba kiküszöbölése lezárhatja a nyitott állományokat. MEGJEGYZÉS: BAM = Block Availability Map (elérhető blokktérkép)
72 **DISK FULL**	Elfogytak a lemezen lévő blokkok vagy a tartalomjegyzék megtelt. A DISK FULL üzenet akkor jelenik meg az 1541-en, amikor még két szabad blokk van, ami lehetővé teszi az adott állomány lezárását..
73 **DOS MISMATCH** (73, CBM DOS V2.6 1541)	A DOS 1-es és 2-es változatai csak olvasni tudják egymás lemezeit. A lemezeket kölcsönösen bármelyik változat tudja olvasni, de az egyik változat által formált lemezre a másik nem

tud írni, mivel a formátum különböző. Az üzenet a nem egyező formátum esetén tett íráskísérletnél jelenik meg. (Egy programrutin lehetővé teszi a formátum átkonvertálását.) Az üzenet bekapcsolás után is megjelenhet.

74 **DRIVE NOT READY**

Kísérlet egy lemezmeghajtó elérésére, melyben nincs lemez.

5. RÉSZ

TEDMON

Bevezetés

A TEDMON egy beépített gépi nyelvű programot jelent, amelynek alkalmazásával könnyen lehet gépi kódú programot írni és futtatni. Tartalma, egy gépi kódú monitor, egy mini-assambler és egy disassambler program. A TEDMON-ban írt programok önállóan is futhatnak, de gyors szubrutinként jól használhatók BASIC programokból is. A PLUS/4 úgy készült, hogy a TEDMON és a BASIC békésen megfér egymással.

A monitor használata

A MONITOR parancs beírása belépteti a TEDMON-t. A gép kiírja a 6502-es regisztereket, és villogtatja a kurzort. Ez jelzi, hogy a gép várja a gépi kódú monitor parancsait.

A parancsok

A: **ASSEMBLE**
Egy assembly programsort visz be.
C: **COMPARE**
összehasonlít két tárterületet, és ha azonosak, kijelzi RETURN-nel .
D: **DISASSEMBLE**
Gépi kódú sorból assembly sort csinál
F: **FILL**
Egy tárterületet feltölt byte-okkal
G: **GO**
Indítja a programot az adott címről
H: **HUNT**
Bizonyos byte-okat megkeres a tárban
L: **LOAD**
Betölt egy file-t szalagról vagy lemezről
M: **MEMORY**
Kiírja a tártartalmat egy adott címtartományban hexadeximális és ASCII állományként
R: **REGISTERS**
Kijelzi a 6502 regiszterek tartalmát.
A regiszterek: PC, SR, AC, XR, YR, SP

S: **SAVE**
A megadott tárrészt szalagra vagy lemezre tárolja.
T: **TRANSFER**
A tártartományt egy kijelölt címtől kezdve átírja egy másik címre.
V: **VERIFY**
Összehasonlít egy file-t a tárból a kazetta vagy a lemez tartalmával
X: **EXIT**
Kilépés a TEDMON-ból, visszatérés BASIC-be.
>: Tártartalmat módosít
:: Módosítja a 6502 regiszterek kiírását.
.: Ugyanaz, mint az A parancs
A $7F8 terület ellenőrzi, hogy a TEDMON a $8000 fölött a ROM-ot vagy a RAM-ot figyeli. Ha ez 0-ra van állítva, a TEDMON kiírja a BASIC-et és a KERNAL-t, amikor „disassembly"-re utasítjuk, vagy arra, hogy képezzen állományt $8000 fölött. Ha $80-ra állítjuk, a TEDMON a BASIC és a KERNAL alatt írja ki a RAM-ot. Ez gyakran szükséges a gépi kódú programok készítéséhez. A $7F8 cím nincs hatással a GO parancsra. A GO parancs elkezdi a program végrehajtását az aktuális helyen (ROM vagy RAM), függetlenül az $7F8 állásaitól.

A TEDMON parancsok leírása

Parancs: A
Cél: Bevinni egy assambly sort
Forma: A <cím> <kód> <változó>
<cím>: hexadecimális szám, amely kijelöli a kód helyét a tárban.
<kód>: standard MOS technológiájú, gépi nyelvű ún. mnemonic kód, pl. LDA, STX stb.
<változó>: az elfogadott címzési módok közül akármelyik lehet. Nullás lapnál két számjegyű hexadecimális szám szükséges, amelynek az értéke $ 100-nál kevesebb. A nullás lapú címekben 4 számjegyű hexadecimális számok kellenek.

RETURN-t használunk akkor, ha az assambly sor végét akarjuk jelezni. Ha valamilyen hiba van a sorban, egy kérdőjelet ír ki, és a következő sorra megy a kurzor. A hiba javítására a képernyőjavító – screen-editor – használható. Miután egy programsort sikerült összeállítani és bevinni, az assambly kiírja a tár következő szabad helyét, így nem kell a sorszámozással foglalkoznunk. Pl.:

```
A 1200 LDX # S00
A 1202
```

Megjegyzés a pont – (.) – azonos az A paranccsal. Pl.: . 1200 ...
Parancs: **C**
Cél: két tárterület összehasonlítása
Forma: C <első cím> <második cím> ...

<első cím>: egy hexadecimális szám, annak a tárterületnek a kezdőcíme. amelyben hasonlítunk.
<második cím>: annak a tárterületnek a hexadecimális zárócíme, amelyben hasonlítunk.
<harmadik cím>: annak a tárterületnek a kezdőcíme, amellyel hasonlítunk. Hexadecimális szám.
Ha a két tárterület azonos, a TEDMON ezt RETURN-nel jelzi. A különböző byte-ok címe mind megjelenik a képernyőn.
Parancs: **D**
Cél: Szétválasztani a gépi kódú sort assembly mnemonic kódra és változóra. Diszesszemblálás.
Forma: D [<cím>][<második cím>]
<cím>: Egy hexadecimális szám, amely meghatározza a szétválasztás kezdőcímét.
<cím>: választható hexadecimális szám, a szétválasztás zárócíme.
A formátuma hasonlít az "A" parancséhoz. Most azonban az első karakter egy pont (.) – az olvashatóság érdekében –, és a hexadecimális kód is kilistázódik. Ez a lista módosítható a "képernyő editor"-ral. A változtatás után a RETURN belépteti a sort és az assembler kész a további módosításokra.
A szétválasztás lapozható, a D beírásával a következő 20 byte a képernyőre kerül. Pl.:

```
D 3000 3004
. 3000 A9 00        LDA $00
. 3002 FF           ???
. 3003 D0 2B        BNE $3030
```

Parancs: **F** (FILL = Betöltés)
Cél: Egy tárterület feltöltése byte-tal.
Forma: F <első cím> <második cím> <byte>

Parancs: **G**
Cél: program indítása egy adott címtől
a) H C 000 FFFF'READ
   Keresi a C000-tól FFFF-ig a READ-et
b) H A 000 A 101 A9 FF 4C
   Keresi a $A9, $FF, és $4C adatokat
   A000-tól A 101-ig

Parancs: **L** (LOAD = betöltés)
Cél: egy file betöltése kazettáról vagy lemezről.
Forma: L <"file neve">,<egység>
<file-név>: bármilyen megengedett file-név lehet
<egység>: hexadecimális szám, amely azt az egységet jelöli ki, ahonnan a betöltés történik.

1: kazetta ; 8: lemez stb.
A tárba töltendő file kezdőcímét a programfile első két byte-ja tartalmazza, azaz a LOAD parancs mindig ugyanarra a helyre tölti be a programot, ahonnan kivettük. Ez igen fontos a gépi kódú programozásban, mivel kevés programot lehet pontosan így visszatölteni. A gép akkor fejezi be a töltést, ha megtalálta az EOF – END OF FILE: program vége – parancsot.
Példák:
L "képernyő", 1
L "játék", 8

Parancs: **M**
Forma: G [<cím>]
<cím>: Választható paraméter, amely a programszámláló (PC) új értékét határozza meg, azt a címet, ahonnan a végrehajtás indul (GO). Ha nem adjuk meg, a végrehajtás a PC aktuális állásától indul. (Ezt akkor láthatjuk, ha egy R parancsot használunk.)
Ez a parancs minden regisztert beállít, és a kijelölt új címen elkezdi a végrehajtást. Figyelmesen használjuk ezt a parancsot! Ha visszatérünk TEDMON-ra egy gépi kódú programból, használjuk a BRK utasítást. Pl.:

```
G 140 C
```

A végrehajtás a $ 140 C címnél kezdődik.

Parancs: **H** (Hunt = keresés)
Cél: A tár egy bizonyos területén végignézni egy bizonyos byte-tartalom előfordulását.
Forma: H <első cím> <második cím> <adatok>
<első cím>: a keresési tartomány kezdőcíme.
<második cím>: a keresési tartomány zárócíme.
<adatok>: az adatkészlet megadható hexadecimálisan vagy ASCII füzérrel. Az ASCII füzérben az első karakter elé egy szimpla idézőjelet írunk; 'READ. Ha az adatok több eleműek és hexadecimálisan adottak, a számokat szóközzel válasszuk el!

Példák:
Cél: A tártartalom kijelzése (Memory display) hexadecimális számként és ASCII állományként egy kijelölt címtartományon belül
Forma: M [<első cím>][<második cím>]
<első cím>: választható paraméter, a tárállomány kezdőcíme. Ha elhagyjuk, egy lapot jelez ki. Az első byte az utolsó kijelölt cím.
<második cím>: a tárállomány zárócíme, választható. Ha nem szerepel, egy lap tartalma jelenik meg. Az első byte az első cím adata.
A tártartalom a következő alakban jelenik meg:
>0310 8B 8C 42 CE 0E CE 4C F4: ... BN.NLI
A tártartalom törölhető képernyőtörléssel. Mozgassuk a kurzort a módosítandó adathoz, javítsuk ki és nyomjuk meg a RETURN-t. Ha rossz RAM szerepel, vagy a ROM módosítására tett kísérletet, hibajelzést (?) ad.

Az adatok ASCII állománya inverzben jelenik meg a hexadecimális adatoktól jobbra. Ha egy karaktert nem lehet kiírni, az inverz pontként (.) jelenik meg. Csakúgy, mint a Dissassambly parancsnál, itt is kérhetünk lapozást úgy, hogy lenyomjuk az M-et, majd a RETURN-t.
Példa:

```
M 1C00
>1C00 41 00 AA AA 00 98 56 45 :A.**..VE
>1C08 41 00 AA AA 00 98 56 45 :A.**..VE
>1C10 41 00 AA AA 00 98 56 45 :A.**..VE
>1C18 41 00 AA AA 00 98 56 45 :A.**..VE
>1C20 41 00 AA AA 00 98 56 45 :A.**..VE
>1C28 41 00 AA AA 00 98 56 45 :A.**..VE
>1C30 41 00 AA AA 00 98 56 45 :A.**..VE
>1C38 41 00 AA AA 00 98 56 45 :A.**..VE
>1C40 41 00 AA AA 00 98 56 45 :A.**..VE
>1C48 41 00 AA AA 00 98 56 45 :A.**..VE
>1C50 41 00 AA AA 00 98 56 45 :A.**..VE
>1C58 41 00 AA AA 00 98 56 45 :A.**..VE
```

Parancs: **>**
Cél: 1-től 8-ig tártartalom módosítása.
Forma: >cím első byte <második byte ... nyolcadik byte>
cím: az első módosításra kerülő tárcím
első byte: a címben rendelt új adat
második byte...: adatok, amelyeket az egymást követő, max. 8 címre kell elhelyezni. Választható paraméter.
Példák:

>2000 08 :	a 2000-es címre 08-at ír.
>3000 23 45 65 :	3000-re 23-at, 3001-re 45-öt és 3002-re 65-öt ír

Parancs: **R**
Cél: á 6502 regiszterek kijelzése. A következő regiszterek tartalma jeleníthető meg: PC – Program Counter; SR – Status Register, AC – Accumulátor; XR YR – index regiszterek; SP – Stach Pointes
Forma: R
Példa: R

```
 PC  SR AC XR YR SP
1002 01  02  03  04  F6
```

Megjegyzés: Regiszterkijelzés módosítására pontosvessző (;) használható, hasonlóan, mint a nagyobb jel (>) a tárregiszterek módosítására.

Parancs: **S** (SAVE = átmentés)
Cél: A tártartalmat szalagra vagy lemezre menti
Forma: S <"file-név">,<egység száma>,<első cím>,<második cím>
<"file-név">: bármely megengedett név.

<egység száma>: 1 jelenti a kazettás egységet, 8 a lemezmeghajtó egységet, ez azonban változhat, ha pl. egynél több meghajtót nasználunk. (Lásd a Lemezmeghajtó kézikönyvet!)
<első cím>: a tárterület kezdőcíme
<második cím>: a tárterület zárócíme + 1. Ennél a címnél levő byte már nem lesz tárolva.
Az így létrehozott program-file első két byte-ja tartalmazza az adatok első címét (tárterület kezdőcíme).
Az L paranccsal hívhatjuk le.
Példa:

S „Játék", 8, 0400, 0C00

A tártartalmat az $ 0400-ás címtől 0C00-ig "Játék" néven lemezre veszi.

Parancs: **T**
Cél: Tártartomány tárainak átírása egyik tárterületről egy másikra
Forma: T <első cím> <második cím> <harmadik cím>

<első cím>: atírandó tartomány kezdőcíme
<második cím>: átírandó tartomány zárócíme
<harmadik cím>: az új terület kezdőcíme
Az adatokat, bármilyen hosszúak is, tetszőlegesen mozgathatjuk, eltolhatjuk mindkét irányba, a megengedett tárterületen belül.
Megjegyzés: a <harmadik cím> nem eshet az első és a második cím közé!
Példa:

T 1401 1600 1400

Eltolja az adatokat az $1401 címtől kezdve az $1600 címig, 1 byte-tal lefelé.

Parancs: **V**
Cél: Egy file-t hasonlít össze a kazettán vagy a lemezen a tártartalommal (VERIFY).
Forma: V <"file-név">,<egység száma>
<"file-név">: tetszőlegesen megengedett név
<egység száma>: hexadecimális szám, azt jelöli, hogy a file melyik egységen van. 01: kazetta, 08 lemez stb. Az összehasonlítás közben a PLUS/4 kiírja: VERIFYING, ha hibát talál kijelzi, ha befejezte visszatér a kurzor.
Példa:
V "Program", 8

Parancs **X**
Cél: visszatérés BASIC-be (EXIT)
Forma: X
Ha a gép X parancsot kap, az SP regiszter visszaáll az aktuális értékre. (R parancs)
A BASIC-be visszatérés után, egy BASIC CLR paranccsal állítsuk vissza a mutatókat!

6. RÉSZ

Képernyőkódok

A következő táblázat a Commodore karakterkészletekben található karaktereket tartalmazza. Ezeket az értékeket kell a képernyőtárba (3072–4095) POKE utasítással bevinni a kívánt karakter megjelenítéséhez. (Ne feledkezzünk meg a színtár beállításáról sem! – 2048–3071) A táblázat az egyes karakterekhez tartozó PEEK értékeket is tartalmazza.

Két karakterkészlet áll rendelkezésre, de egyidejűleg nem érhetők el. Ez azt jelenti, hogy miközben az egyik készlet karakterei láthatók a képen nem jeleníthetünk meg a másik készletből származó karaktereket. A karakterkészletek átkapcsolása a SHIFT és C= billentyűk egyidejű lenyomásával lehetséges.

BASIC-ből a PRINT CHR$(142) utasítással válthatunk a nagybetűs/grafikus üzemmódra, míg a PRINT CHR$(14) utasítással a nagybetűs/kisbetűs üzemmódra.

A táblázatban taláiható bármely értéket inverz módban is megjeleníthetjük. Az inverz kódokat a táblázatból vett értékeknek és 128-nak az összeadásával nyerhetjük.

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	T	t	20			40
A	a	1	U	u	21	)		41
B	b	2	V	v	22	*		42
C	c	3	W	w	23	+		43
D	d	4	X	x	24	,		44
E	e	5	Y	y	25	–		45
F	f	6	Z	z	26	.		46
G	g	7	[		27	/		47
H	h	8	£		28	0		48
I	i	9	]		29	1		49
J	j	10	↑		30	2		50
K	k	11	←		31	3		51
L	l	12	SPACE		32	4		52
M	m	13	!		33	5		53
N	n	14	"		34	6		54
O	o	15	#		35	7		55
P	p	16	$		36	8		56
Q	q	17	%		37	9		57
R	r	18	&		38	:		58
S	s	19	'		39	;		59

A 128–255-ig terjedő kódok a 0–127-ig terjedő kódok inverz megfelelői.

SET 1	SET 2	POKE
<		60
=		61
		62
?		63
		64
	A	65
	B	66
	C	67
	D	68
	E	69
	F	70
	G	71
	H	72
	I	73
	J	74
	K	75
	L	76
	M	77
	N	78
	O	79
	P	80
	Q	81
	R	82
	S	83

SET 1	SET 2	POKE
	T	84
	U	85
	V	86
	W	87
	X	88
	Y	89
	Z	90
		91
		92
		93
		94
		95
SPACE		96
		97
		98
		99
		100
		101
		102
		103
		104
		105
		106
		107

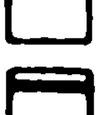
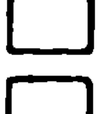
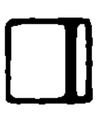
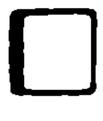
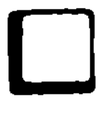
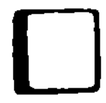
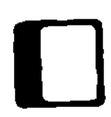
SET 1	SET 2	POKE
		108
		109
		110
		111
		112
		113
		114
		115
		116
		117
		118
		119
		120
		121
		122
		123
		124
		125
		126
		127

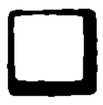
7. RÉSZ.

A következő táblázat megmutatja, hogy a PRINT CHR$(X) utasításra milyen karakterek jelennek meg a képernyőn. Az X 0–255-ig változhat.
Kiolvashatjuk azt is, hogy a PRINT ASC("X") utasítás milyen értékeket ír ki, hogy X bármelyik karakter lehet a billentyűzeten.
Jól használható a táblázat a GET utasításban kapott karakterek kiértékeléséhez, a nagybetű/kisbetű átalakításához, a karakterkódokra vonatkozó parancsokhoz olyan esetben, ha a karakter nem írható idézőjelek közé

ASCII és CHR$ kódok

KIIRÁS	CHR$	KIIRÁS	CHR$	KIIRÁS	CHR$	KIIRÁS	CHR$
	0	↓	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLEAR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
fehér	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41		58
hatástalanítja a C= váltót	8		25	*	42	;	59
aktivizálja a C= váltót	9		26	+	43	<	60
	10	ESCAPE	27	,	44	=	61
	11	vörös	28	-	45	>	62
	12	→	29		46	?	63
RETURN átkapcsolás kisbetűkre	13	zöld	30	/	47	@	64
	14	kék	31	0	48	A	65
	15	helyköz	32	1	49	B	66
	16		33	2	50	C	67

KIIRÁS	CHR$	KIIRÁS	CHR$	KIIRÁS	CHR$	KIIRÁS	CHR$
D	68		97		126	világoszöld	155
E	69		98		127	PUR	156
F	70		99		128		157
G	71		100	narancs	129	YEL	158
H	72		101	villogás be	130	CYN	159
I	73		102		131	SPACE	160
J	74		103	villogás ki	132		161
K	75		104		133		162
L	76		105		134		163
M	77		106		135		164
N	78		107		136		165
O	79		108		137		166
P	80		109		138		167
Q	81		110		139		168
R	82		111		140		169
S	83		112	SHIFT RETURN átkapcsolás nagybetűkre	141		170
T	84		113		142		171
U	85		114		143		172
V	86		115	fekete	144		173
W	87		116		145		174
X	88		117	RVS OFF	146		175
Y	89		118	CLEAR HOME	147		176
Z	90		119	INST DEL	148		177
[	91		120	barna	149		178
£	92		121	sárga/zöld	150		179
]	93		122	rózsaszín	151		180
↑	94		123	kék/zöld	152		181
←	95		124	világoskék	153		182
	96		125	kék	154		183

184

186

188

190

185

187

189

191

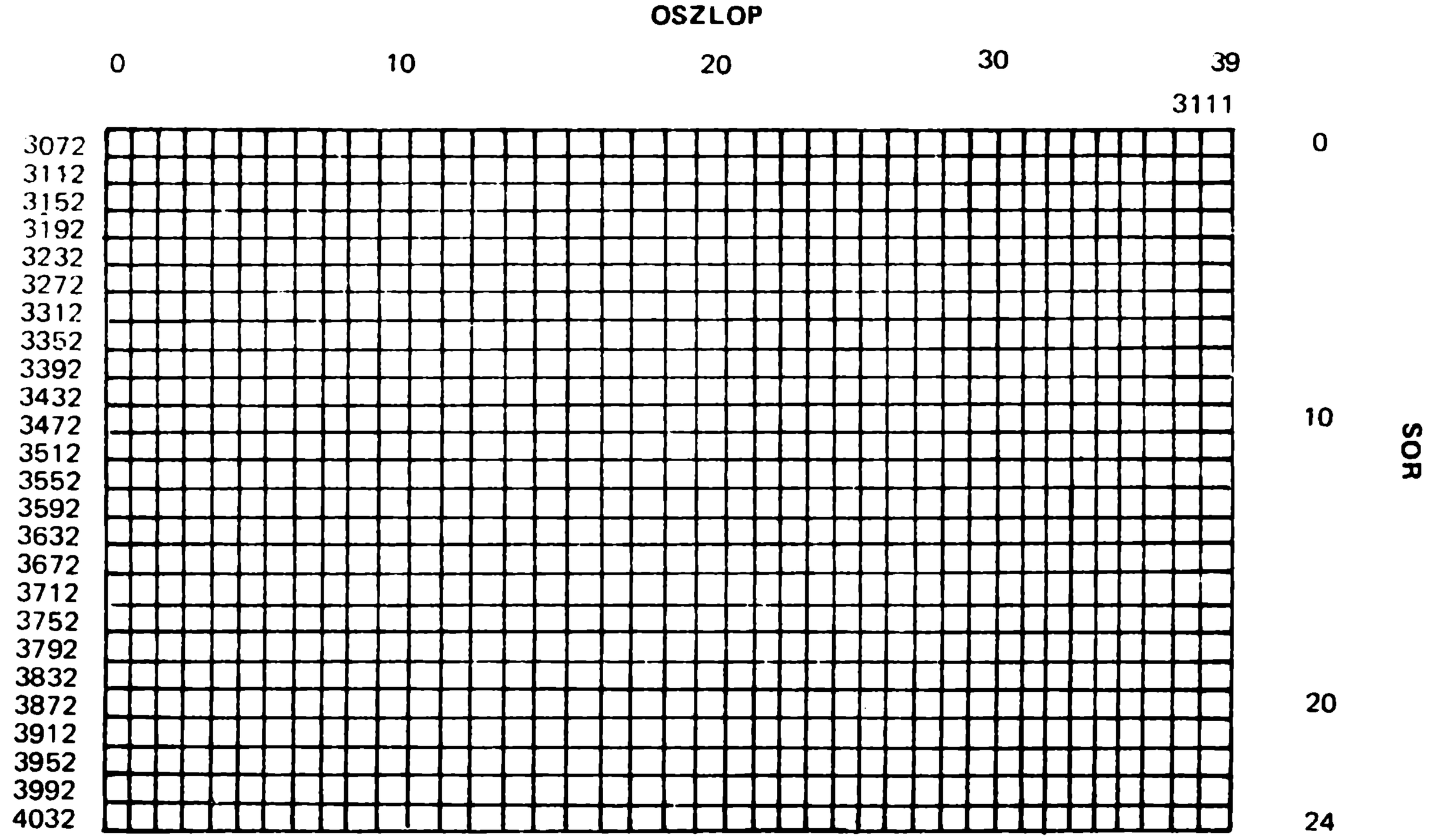
CODES	**192-223**	**UGYANAZ**	**96-127**
CODES	**224-254**	**UGYANAZ**	**160-190**
CODE	**255**	**UGYANAZ**	**126**

8. RÉSZ

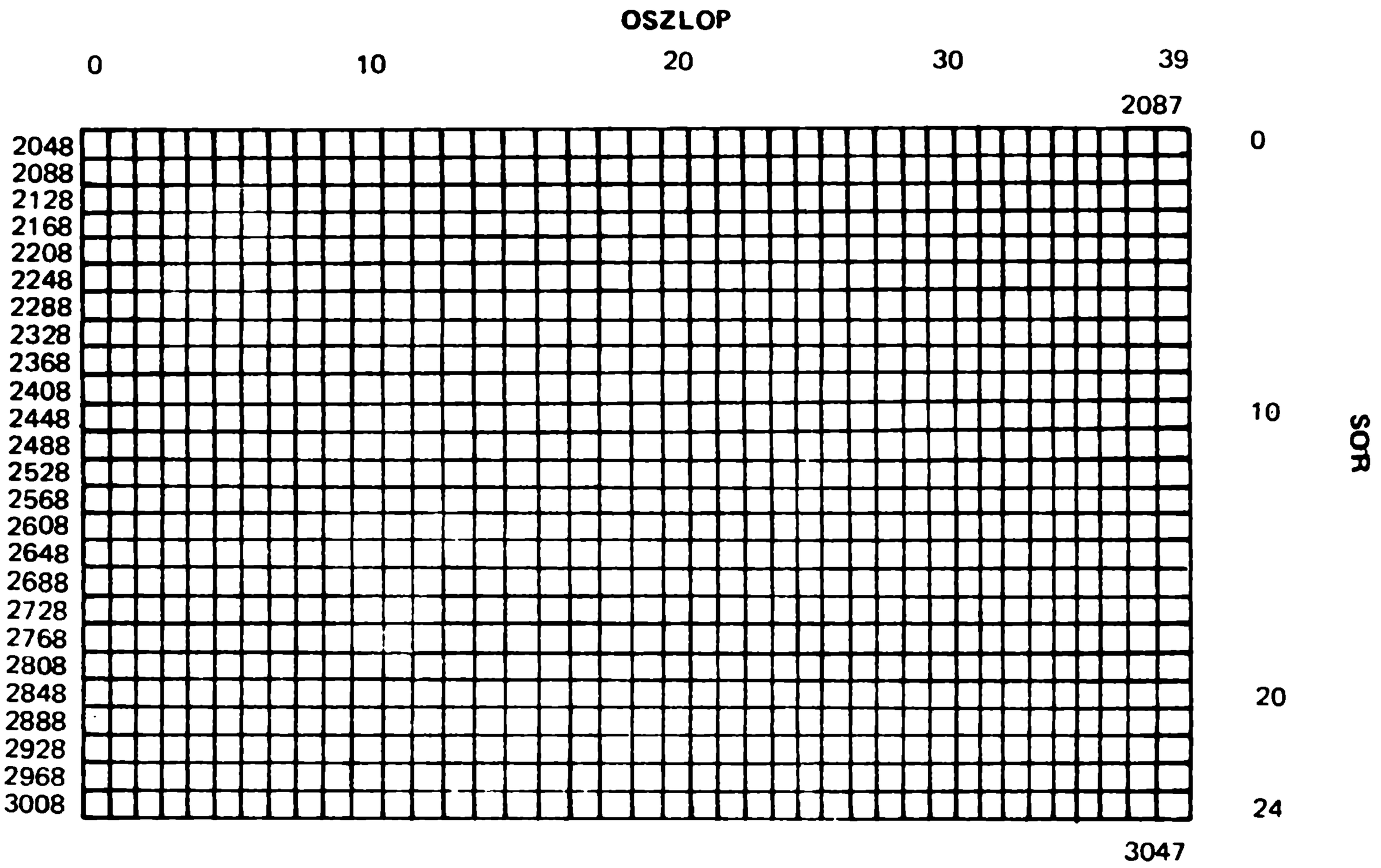
Képernyőtár és színtár térképek

A következő táblázat azt mutatja meg, hogy a képernyőn elhelyezett karakterek pozíciója milyen tárcímen található. A másik táblázat azt tartalmazza, hogy az egyes képernyőkarakterek színe milyen címen van. Ezek ismeretében könnyen megváltoztathatjuk a képernyő tartalmát közvetlenül is. (Lásd POKE utasítás.)

Képernyőtár térkép



Szintár térkép



A karakterek színét szabályozó értékek

0 FEKETE	8 NARANCS
1 FEHÉR	9 BARNA
2 PIROS	10 SÁRGÁSZÖLD
3 ENCIÁN	11 RÓZSASZÍN
4 BÍBOR	12 KÉKESZÖLD
5 ZÖLD	13 VILÁGOSKÉK
6 KÉK	14 SÖTÉTKÉK
7 SÁRGA	15 VILÁGOSZÖLD

Egy szín élességét úgy választjuk ki, hogy az élességszín (0–7) értékét 16-tal megszorozzuk, és hozzáadjuk a színértékhez. A karakter villogni fog, ha a színértékét 128-cal növeljük.

9. RÉSZ

Tárregiszter térkép

Cím	Tartalom	Megjegyzés
$FFFE–FFFF	IRQ vektor	
$FFFC	RES vektor	
$FFA	NMI vektor (nem használatos)	ROM magas (folytatás)
$FF81–FFF5	KERNAL ugrótáblázat	
$FF40–FF80		
$FF00–FF3F	TED CHIP	
$FE00–FEFF	DMA lemezrendszer	TED chip és bevitel/kivitel terület megjelennek minden tár térképén
$FDF0–FDEF		
$FDD0–FDDF	Modul port (kapu)	
$FD10–FDF1	6529 párhuzamos port	
$FD00–FD0F	ACIA	
$FCD0–FCFF		ROM rutinok megjelennek minden ROM térképén
$D800–FCFF		
$D000–D7FF	Karakter ROM	ROM magas
$C000–D7FF	More BASIC	
$8000–BFFF	BASIC	ROM alacsony
$4000–FFFF	ROM, egyben a BASIC szövegmező kezdete amikor finomfelbontású grafikát használunk	
$2000–3FFF	Bit térkép képernyőadatok	
$1C00–1FFF	Nagypontosságú képernyő videó mátrix	
$1800–1BFF	Nagypontosságú képernyő kiegészítő/járulékos byte-ok	
$1000–	BASIC szövegmező (tártérkép ki)	
$0C00–OFFF	Szöveg videó mátrix (képernyőtár)	
$0800–0BFF	Szövegjárulékos byte-ok (színtár)	
$0000–0FFF	Rendszertároló	

* Megjegyzés: a 64k RAM rendszerben a RAM $0000–$FCFF-ig terjed, és $FF40–$FFFF-ig.

10. rész

Matematikai függvények kiszámítása

A BASIC 3.5-ben nem definiált függvényeket a következőképpen lehet kiszámítani

Függvény	BASIC megfelelő
Szekáns	SEC(X) = /COS(X)
Koszekáns	CSC(X) = 1/SIN(X)
Kotangens	COT(X) = 1/TAN(X)
Inverz szinusz	ARCSIN(X) = = ATN(X/SQR(−X*X+1))
Inverz koszinusz	ARCCOS(X) = −ATN(X/SQR (−X*X+1))+Π/2
Inverz szekáns	ARCSEC(X) = ATN(X/SQR(X*X−1))
Inverz koszekáns	ARCCSC(X) = ATN(X/SQR(X*X−1)) +(SGN(X)−1*Π/2)
Inverz kotangens	ARCOT(X) = ATN(X)+Π/2
Hiperbolikus szinusz	SINH(X) = (EXP(X)−EXP(−X))2
Hiperbolikus koszinusz	COSH(X) = (EXP(X)−EXP(−X))2
Hiperbolikus tangens	TANH(X) = EXP(−X)/(EXP(X)+EXP (−X))*2+1
Hiperbolikus szekáns	SECH(X) = 2/(EXP(X)+EXP(−X)
Hiperbolikus koszekáns	CSCH(X) = 2/(EXP(X)−EXP(−X)
Hiperbolikus kotangens	COTH(X) = EXP(−X)/(EXP(X) −EXP(−X))*2+1
Inverz hiperbolikus szinusz	ARCSINH(X) = = +LOG(X+SQR(X*X+1))
Inverz hiperbolikus koszinusz	ARCCOSH(X) = = +LOG(X+SQR(X*X−1))
Inverz hiperbolikus tangens	ARCTANH(X) = = LOG((1+X)/(1−X))/2
Inverz hiperbolikus szekáns	ARCSECH(X) = LOG((SQR (−X*X+1)+1)/X)
Inverz hiperbolikus koszekáns	ARCCSCH(X) = LOG((SGN(X)*SQR (X*X+1)/X))
Inverz hiperbolikus kotangens	ARCCOTH(X) = = LOG((X+1)/(X−1))/2

11. RÉSZ

ZENEI HANGJEGYTÁBLÁZAT

Hangjegy	Hangregiszter-érték	Aktuális frekvencia; Hz
A	7	110
B	118	123.5
C	169	130.8
D	262	146.8
E	345	164.7
F	383	174.5
G	453	195.9
A	516	220.2
B	571	246.9
C	596	261.4
D	643	293.6
E	685	330
F	704	349.6
G	739	392.5
A	770	440.4
B	798	494.9
C	810	522.7
D	834	588.7
E	854	658
F	864	699
G	881	782.2
A	897	880.7
B	911	989.9
C	917	1045
D	929	1177
E	939	1316
F	944	1398
G	953	1575

A fenti táblázat 4 oktávnyi hang hangregiszterértékeit tartalmazza. Ezeket az értékeket a SOUND utasítás második paramétereként használjuk. Ha a táblázat első hangját használjuk (A – regiszterérték 7), akkor a 7-et használjuk 2. számként a SOUND parancs mögött: SOUND 1,7,30.

A hangerőállításhoz először a VOL 8-at használjuk. Ha olyan frekvenciákat akarunk megkeresni, amelyek nincsenek a táblázatban, akkor a következő formát használjuk:

Hangregiszter érték = 1024 – (111 860.781 / frekvencia). Mind a hangregiszter értéktáblázat, mind a fenti formák az NTSC televíziókra érvényesek. Ez az a tv, amelyet általánosan használnak az USA-ban és Kanadában. Ha az Ön országában a PAL rendszerű a tv, a következő formát kell használnia, hogy az egész táblázatra érvényesen ki tudja számolni a hangregiszter értékeket:

Hangregiszter érték = 1024 – (111 840.45/frekvencia)

12. RÉSZ

A) KIPRÓBÁLANDÓ PROGRAMOK

```
5 GRAPHIC 3, 1: GRAPHIC 0, 1
10 INPUT "KITAKARITSAK-E"; A$
20 INPUT "FOROGJAK-E"; B$
30 INPUT "MEGVALTOZTASSAM A MOZGASOM"; C$
40 INPUT "INDULJAK-E" ; D$
50 IF A$="Y" THEN DIM A (3,200)
60 DEF FNA (X) = INT(RND(1) * X)
70 IF D$="Y" THEN X1=FNA(80)+80: X2=FNA(80)+80:
   Y1=FNA(100)+100
75 IF D$="Y"THEN Y2=FNA(100)+100
80 IF D$<>"Y" THEN X1=80: X2=80: Y1=100: Y2=100
90 GRAPHIC 3: FOR L=1 TO 3: COLOR L, FNA(15)+2,FNA(8):
   NEXT
100 IF C1<1 THEN COLOR FNA(3)+1,FNA(15)+2,FNA(8):
    C1=FNA(40)+20
110 IF C2<>0 THEN 140: ELSE XA=FNA(11)–5: XB=FNA(11)–5:
    YA=FNA(13)–6
115 YB=FNA(13)–6
120 IF C$="Y" THEN C2=FNA(10)+5
130 IF B$="Y" THEN XB=–XA: YB=–YA
140 IF C3<1 THEN C=FNA(3)+1: C3=FNA(10)
145 IF A$="Y" THEN DRAW 0, A(0,P),A(1,P) TO A(2,P),A(3,P)
150 X1 = X1 + XA: X2= X2+ XB: Y1= Y1+ YA: Y2= Y2+ YB
160 IF X1<0 OR X1>159 THEN XA= –XA: X1= X1+XA
170 IF X2<0 OR X2>159 THEN XB= –XB: X2= X2+XB
180 IF Y1<0 OR Y1>199 THEN YA= –YA: Y1= Y1+YA
190 IF Y2<0 OR Y2>199 THEN YB= –YB: Y2= Y2+YB
200 DRAW C, X1, Y1 TO X2, Y2
210 IF A$="Y" THEN A(0,P)= X1: A(1,P)=Y1: A(2,P)=X2:
    A(3,P)=Y2: P= P+1
215 IF A$="Y" THEN IFP>200THENP=0
220 C1=C1–1: C2=C2–1: C3=C3–1: GOTO 100
```

B) HANGHATÁSOK

1. Farkasüvöltés

```
10 VOL7
20 FORL=400TO800STEP20
30 SOUND1,L,3:NEXT
40 FORL=300TO600STEP40
50 SOUND1,L,3:NEXT
60 FORL=600TO300STEP-40
70 SOUND1,L,3:NEXT
```

2. Komputermániákus

```
10 VOL7
20 FORL=1TO100
30 SOUND1,INT(RND(0)*500)+400,4
40 NEXT
```

3. Telefon

```
10 VOL7
20 FORL=1TO5
30 FORM=1TO60
40 SOUND1,466,1
50 SOUND1,1020,1
60 NEXT
70 FORZ=1TO2000:NEXT
80 NEXT
```

4. Foglaltjelzés

```
10 VOL7
20 FORL=1TO15
40 SOUND1,466,20
50 SOUND1,1020,15
80 NEXT
```

5. Buborék

```
10 VOL7
20 GRAPHIC1,1
30 FORM=1TO50
40 GOSUB80
50 SOUND1,900-R*20,(YR/2)+50
60 CIRCLE1,X,Y,R,YR
70 NEXT:GRAPHIC0:END
80 X=INT(RND(0)*280)+20
90 Y=INT(RND(0)*160)+20
100 R=INT(RND(0)*40)+5
110 YR=R/1.3
120 RETURN
```

6. Szivattyú

```
10 VOL7
20 FORM=1TO 20
30 FORL=900TO850STEP-10
40 SOUND1,L,1

50 NEXT
60 FORL=850TO900STEP10
70 SOUND1,L,1
80 NEXT
100 NEXT
```

7. Zenei sorok

```
10 VOL7
15 X1=0,Y1=0
20 GRAPHIC1,1
30 GETA$:IFA$=""THENGRAPHICO:END
40 GOSUB80
45 FORL=1TODSTEP2
50 SOUND1,X*3,5
65 SOUND2,Y*3,5
60 DRAW1,X,Y
65 X=X+2*DX:Y=Y+2*DY
70 NEXT:GOTO30
80 X=X1:X1=INT(RND(0)*280)+20
90 Y=Y1:Y1=INT(RND(0)*160)+20
100 A=X1-X:B=Y1-Y:D=SOR(A*A+B*B)
110 DX=A/(D+1):DY=B/(D+1)
120 RETURN
```

13. RÉSZ

RS–232 INTERFACE

BEVEZETÉS

A COMMODORE PLUS/4 a beépített RS–232 interface-en keresztül csatlakoztatható bármely RS 232 modemen különböző eszközökhöz, pl. nyomtatókhoz. Az összekapcsoláshoz azonban a kábelen kívül némi kiegészítő ismeretek is szükségesek a programozásról.
Az interface standard, de a feszültség TTL rendszerű (0–5V) és nem a normál −12 V és +12 V közötti tartomány. Az összekötő rendszernek rendelkeznie kell a szükséges feszültségátalakítóval. (Az RS–232 interface erre alkalmas.)

Az RS–232 interface szoftver BASIC-ből vagy a KERNAL rutinból elérhető a gépi kódú programozáshoz. Ez a rész bemutatja, hogy BASIC-ből hogyan használható az RS–232. Részletesebb információk a COMMODORE PLUS/4 Programozók Kézikönyvében találhatók, a részletes gépi kódú programozási ismeretekkel együtt.
A szokásos BASIC parancsokat használhatjuk:
OPEN, CLOSE, CMD, INPUT#, GET#, PRINT# és az ST inverz változót.
Az INPUT# és a GET# a pufferből veszi az adatokat, a PRINT# és a CMD pedig az átalakítóban helyezi el azokat.

Az RS–232 csatorna megnyitása

Egyszerre mindig egy csatorna lehet nyitva!
A file-mezőbe maximum négy karakter küldhető. Az első kettő a vezérlő és utasítás regiszter karaktere, a másik kettő a későbbi rendszerválasztásra alkalmas.
BASIC sintaxis:

OPEN logikai file, 2, 0<"> Control command regiszter <">

Pl.:

OPEN 2,2,0 ,CHR$(5) + CHR$ (15)

Logikai file: 1–255. Ha > 127, a RETURN-t kiírás is követi.
Control regiszter: lásd az ábrát, egy byte-os karakter
Command regiszter: lásd az ábrát, egy byte-os karakter

Adatok az RS–232 csatornából

A COMMODORE PLUS/4 127 karakternyi adatot fogadhat el, túlcsordulás nélkül, amit a BASIC-ből az ST változó jelez. Ezután már minden további beérkező karakter elvész. Éppen azért érdemes a fogadó puffert a lehető legüresebben tartani. Viszont a BASIC-ben lassan megy az adatmozgatás, így ez a művelet már okvetlenül gépi kódú programot igényel:

Control-regiszter

7	6	5	4	3	2	1	0

Bitek STOP

0 = 1 bit STOP

1 = 2 bit STOP

1 bit STOP, ha a szóhossz = 8 bit és paritásbit *

$1\,^1/_2$ bit STOP ha a szóhossz = 5 bit és nincs paritás-bit

Szóhossz

Bit 6	Bit 5	Adatszó hossza
0	0	8
0	1	7
1	0	6
1	1	5

Óra

0 = külső óra

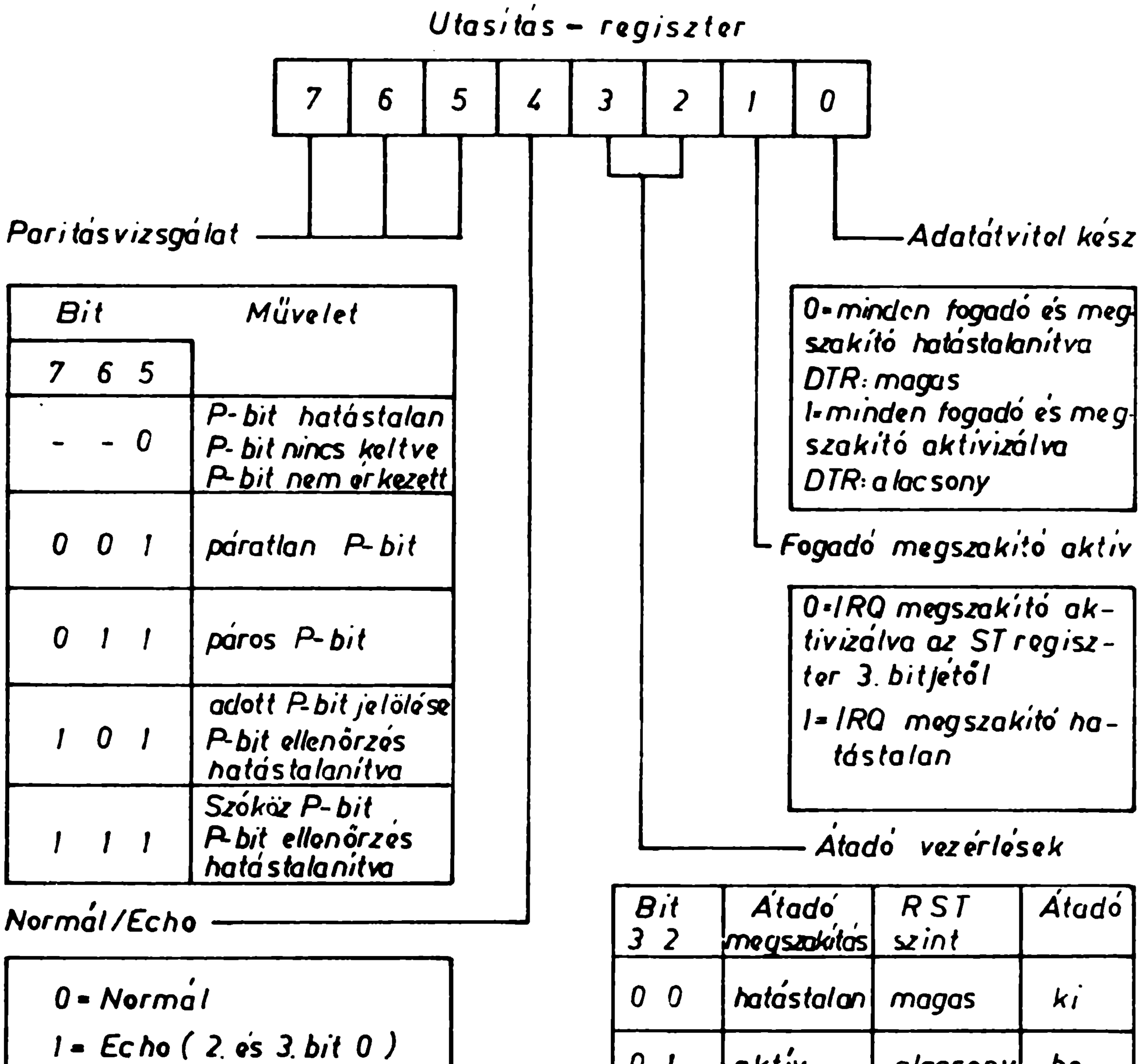
1 = adatatvitel

3	2	1	0	adatátvitel (baud/s)
0	0	0	0	16x külső óra
0	0	0	1	50
0	0	1	0	75
0	0	1	1	109.92
0	1	0	0	134.58
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	1800
1	0	1	0	2400
1	0	1	1	3600
1	1	0	0	4800
1	1	0	1	7200
1	1	1	0	9600
1	1	1	1	192000

	7	6	5	4	3	2	1	0
Hardver Reset	0	0	0	0	0	0	0	0
Pogram Reset	-	-	-	-	-	-	-	-

* Ez engedi a 9. bit átvitelét (8. bit + paritás)

BASIC forma:
Ajánlott parancs: GET # logikai file, sztringváltozó
Nem ajánlott parancs: INPUT # logikai file, változólista



Bit 7 6 5	Művelet
- - 0	P-bit hatástalan P-bit nincs keltve P-bit nem érkezett
0 0 1	páratlan P-bit
0 1 1	páros P-bit
1 0 1	adott P-bit jelölése P-bit ellenőrzés hatástalanítva
1 1 1	Szóköz P-bit P-bit ellenőrzés hatástalanítva

Bit 3 2	Átadó megszakítás	RST szint	Átadó
0 0	hatástalan	magas	ki
0 1	aktív	alacsony	be
1 0	hatástalan	alacsony	be
1 1	hatástalan	alacsony	BRK

	7	6	5	4	3	2	1	0
Hardver Reset	0	0	0	0	0	0	0	0
Program Reset	-	-	-	0	0	0	0	0

Megjegyzés: Ha a szóhossz rövidebb 8 bitnél, a fel nem használt bitek értéke 0 lesz.
Ha a GET# utasítás nem talál adatot, akkor a nulla karaktert adja meg (" ").
Ha az INPUT# utasítást használjuk, a rendszer addig vár, amíg egy nem-nulla karaktert és RETURN-t nem kap. Így a rendszer könnyen RESET állapotba kerülhet, ezért nem ajánlatos az INPUT# és CHRIN rutinokat használni.

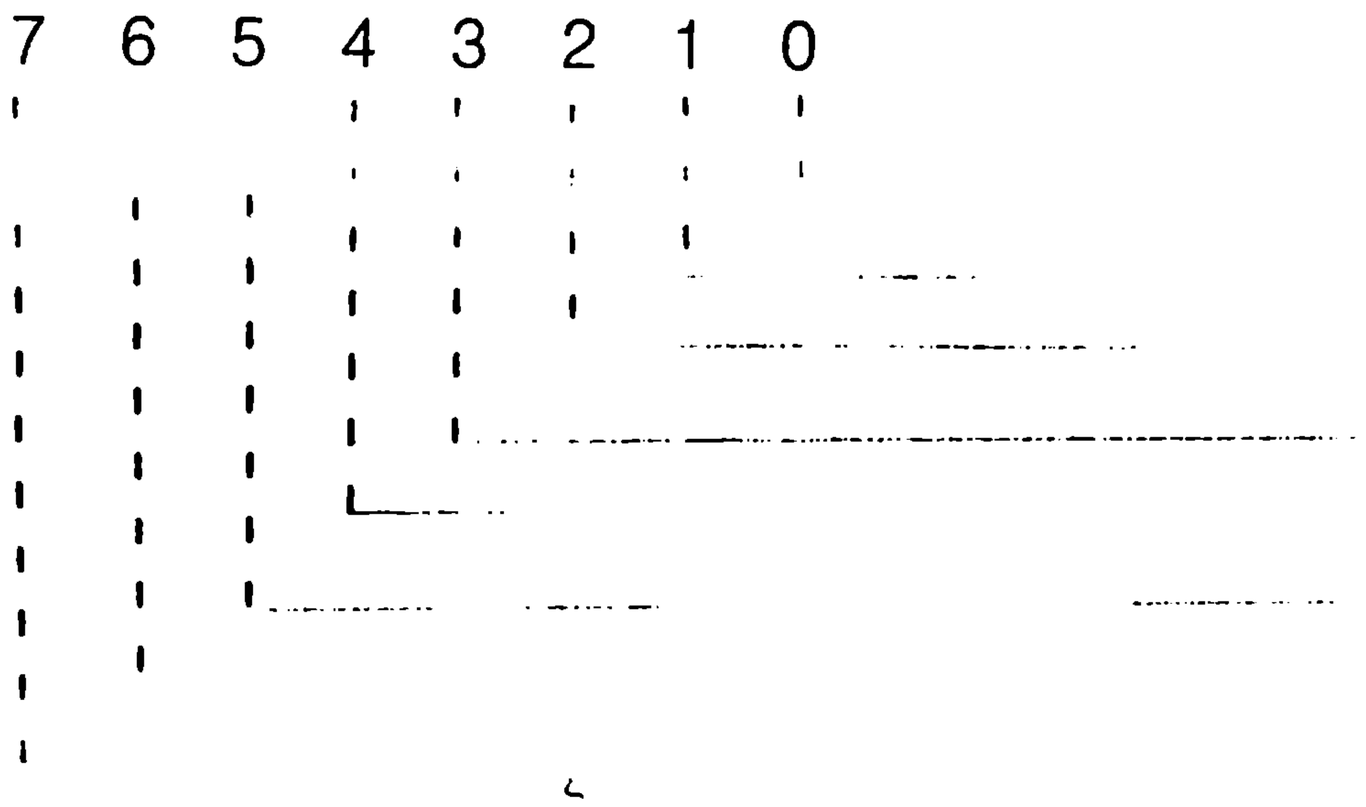
Adatok az RS–232 csatornára

BASIC forma:
CMD logikai file – ugyanúgy működik, mint a BASIC specifikációban PRINT# logikai file, változólista

Az RS–232 adatcsatorna lezárása

Lezárva egy RS–232 file-t minden adat törlődik, minden RS–232 adatcsere leáll, RTS és Sout értéke 0 lesz.
BASIC forma:
CLOSE logikai file
RS–232 portok

ID	leírás	E/A	ABV	Ki
C	kapott adatok	BB	Sin	Be
D	elküldendő kérés	CA	RTS	Ki
E	adatterminál kész	CD	DTR	Ki
F	ring indikátor	CE	RI	Be
H	fogadott vonal jel	CF	DCD	Be
J	nincs hozzárendelés		XXX	Be
K	törlés elküldésre	CB	CTS	Be
L	adatállomány kész	CC	DSR	Be
B	kapott adatok	BB	Sin	Be
M	átadott adatok	BA	Sout	Be
A	védőföldelés	AA	GND	
N	Belső földelés	AB	GND	



RS–232 ST-regiszter

Megjegyzés: Ha a bit = 0, nem talált hibát.

Az RS regisztert BASIC-ből az ST változóval lehet beolvasni. Ha a BASIC olvassa az ST-t, vagy a KERNAL READST rutint használjuk, akkor kilépéskor az RS regiszter tartalma törlődik. Ha többször használjuk, ki kell menteni egy másik változóba. Pl.: SR = ST: REM ST AZ SR-HEZ RENDELVE

Az RS regiszter tartalmát csak akkor olvassuk be (töröljük), ha az RS–232 csatorna volt az utoljára használt I/0 egység.

BASIC nyelvű mintaprogram

"Telekommunikációs csatorna megnyitása"

```
100 OPEN 5,2,2, CHR$/22/ + CHR$/5/: REM Pufferkijelölés és csatornanyitás
110 DIM F%(255), T%(255)
120 FOR J= 32 TO 64: T%(J) = J:NEXT
130 T%(13) = 13: T%(20) = 8: RV = 18: CT = 0
220 FOR J = 65 TO 90: K = J + 32: T%(J) = K: NEXT
230 FOR J = 91 TO 95: T%(J) = J: NEXT
240 FOR J = 193 TO 218: K = J–128: T%(J) = K: NEXT
250 T%(146) = 16: T%(133) = 16
255 T%(137) = 3: T%(134) = 17: T%(138) = 19
260 FOR J = 0 TO 255
270 K = T%(J)
280 IF K < > 0 THEN F%(K) = J: F%(K + 128) = J
290 NEXT
300 PRINT" "CHR$(147)
310 GET # 5,A$
320 IF A$ = "" THEN 360
330 PRINT" "CHR$(157);CHR$(F%(ASC(A$)));
340 IF F%(ASC(A$)) = 34 THEN PRINT CHR$(27)"O";
350 GOTO 310
360 PRINT CHR$(RV)" "CHR$(157);CHR$(146);: GET A$
370 IF A$ < > "" THEN PRINT # 5,CHR$(T%(ASC(A$)));
380 CT = CT + 1
390 IF CT = B THEN CT = 0: RV = 164–RV
410 GOTO 310
```

TÁRGYMUTATÓ

A
Adatmozgatás 48
ASCII kódok 139
AUTO parancs 82

B
BACKUP parancs 82
BASIC ismerettár 79
BASIC rövidítések 119
Beépített szoftver 22
Billentyűzet 15
Botkormány 11
BOX utasítás 66, 90

C
C= billentyű 15, 17
Cartridge-modul 22
CHAR utasítás 65, 91
CHR $ karakterfüggvény 113
CHR $ kódok 139
CIRCLE utasítás 66, 91
CLEAR/HOME billentyű 16
CLOSE utasítás 92
CLR utasítás 92
CMD utasítás 92
COLLECT parancs 83
COLOR utasítás 61, 92
CONT parancs 83
CONTROL billentyű 16
COPY parancs 83

D
DATA utasítás 50, 93
DEF FN utasítás 45, 93
DELETE parancs 84
DIM utasítás 93, 116
DIRECTORY parancs 19, 26, 84
DLOAD parancs 19, 25, 84
DO-LOOP utasítás 52, 94
DRAW utasítás 64, 95
DSAVE parancs 19, 26, 85

E
END utasítás 95
ESC billentyű 18, 99

F
Finomfelbontású grafika 62
FLASH ON/OFF 17
FOR...TO...NEXT utasítás 51
FOUND kijelzés 23
Függvények 45, 110

G
GET utasítás 96
GET# utasítás 96
GETKEY utasítás 50, 97
GOSUB utasítás 54, 97
GOTO utasítás 51, 97
Grafikus karakterek 15, 17, 57
Grafikus kurzor (PC) 90, 91, 99, 109
GRAPHIC utasítás 62, 97
GSHAPE utasítás 106

H
Hanggenerátor 72
HEADER parancs 25, 85
HELP 20, 85
Hibaelhárítás 32
Hibajavítás 32
Hibaüzenetek 124

I
IF...THEN utasítás 53, 98
INPUT utasítás 98
INPUT# utasítás 99
INST/DEL billentyű 16
Inverz kijelzés 25

K
Karaktermozgatás 59
Kazetta 23

Képernyőablak 37
Képernyőkódok 136
Képernyőtörlés 36
KEY 19, 86
Kurzor 12

L
LEFT $ karakterfüggvény 114
Lemezformálás 25
Lemez-hibaüzenetek 126
Lemezmeghajtó 24
LEN $ karakterfüggvény 114
LET utasítás 99
LIST parancs 86
LOAD parancs 87
LOCATE utasítás 64, 99, 109

M
Mágneslemez 24
MID $ karakterfüggvény 114
MONITOR utasítás 99
Műveletek 40, 118
Műveletek prioritása 42, 118

N
NEW parancs 87
NEXT utasítás 51, 95, 100
Nyomtató 12

O
On utasítás 100
ON/OFF kapcsoló 9
OPEN parancs 100

P
PAINT utasítás 68, 101
Parancsok 82
PEEK függvény 111
Perifériák 12
Pixel kurzor lásd Grafikus kurzor
POKE utasítás 101
PRINT utasítás 48, 101
PRINT# utasítás 102
PRINT USING utasítás 102
Programok tárolása 19, 24, 83. 85, 88
PUDEF utasítás 104

R
READ utasítás 50, 104
Relációjelek 39
REM utasítás 54, 104
RENAME parancs 87
RENUMBER parancs 88
RESET 9
RESTORE utasítás 105
RESUME utasítás 105
RETURN billentyű 15
RETURN utasítás 54, 97, 105
RIGHT $ karakterfüggvény 114
RND függvény 44, 112
RS–232 csatorna interface 151
RS–232 user-port 151
RUN parancs 88
RUN/STOP billentyű 15
RVS ON/OFF billentyű 17

S
SAVE parancs 88
SCALE utasítás 105
SCNCLR utasítás 105
SCRATCH parancs 89
SHIFT billentyű 15
SOUND utasítás 72, 106
SSHAPE utasítás 106
STOP uatsítás 107
SYS utasítás 107
Színbillentyűk 17, 29, 61, 68
Szoftver 12, 19
Szubrutin 53

T
TAB függvény 60
Tárbővítő 11
TEDMON
Többszínű grafika 69
Tömbök 93, 116
TRAP utasítás 107

TROFF utasítás 107
TRON utasítás 107
TV 10

V

Változók 42, 115
VERIFY parancs 89
VOL utasítás 72, 108

W

WAIT utasítás 108

Z

Zene 72
Zenei hangjegy táblázat 146

99,- Ft